ABB Robotics

Application manual
Spot options

Power and productivity
for a better world™

Application manual
Spot options

3HAC024762-001
Revision H
RobotWare 5.14

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

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

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

Additional copies of this manual may be obtained from ABB at its then current charge.

ABB AB
Robotics Products
SE-721 68 Västerås
Sweden

1 Summary

1.1 Spot welding

The Spot options

The *Spot* options are general and flexible software platforms for creation of customized and easy to use function packages for different types of spotweld systems and process equipments.

There are three different Spot options supporting spotweld, two with servo guns and one with pneumatic guns.

- The option *Spot* provides support for sequential welding with one or several pneumatic gun equipments, but also welding with four pneumatic guns at the same time.
- The option *Spot Servo* provides support for sequential welding with one or several servo gun equipments, but also welding with four servo guns at the same time.
- The option *Spot Servo Equalizing* has the same functionality as *Spot Servo* but can also be used for guns without mechanical equalizing systems.

All *Spot* options provides dedicated spotweld instructions for fast and accurate positioning combined with gun manipulation, process start and supervision of the different gun equipments.

Communication with the welding equipment is done with digital inputs and outputs.

Note that the *Spot* options are general and can be extensively customized. They have a default ready to use functionality after installation but should be customized by changing configuration data, RAPID data, and RAPID routines.

1.2 Spot welding features

Features

The Spot Options package contains the following features:

- Fast and accurate positioning using the unique QuickMove and TrueMove concept.
- Gun pre-closing, that is having the gun closing synchronized with weld position.
- Gun equalizing, that is. having the gun “floating” around the weld position.
- Constant tip force during welding for servo guns.
- Manual actions for welding and gun control.
- Several simulation possibilities for test purposes.
- Reverse execution with gun control.
- Weld error recovery with automatic rewelding.
- User-defined supervision and error recovery.
- User-defined autonomous supervision, such as weld current signal and water cooling start and continuous supervision of the water flow.
- Wide customizing possibilities.
- Welding with four guns at the same time.
- Software equalizing functions (if the Spot Servo Equalizing option is installed).
- Default “ready to use” functionality directly after installation.
- Detect missing or improper plates for servo guns.
- Gun calibration functions for servo guns.
- Gun data, such as weld counters and tip wear data, for each used gun.
- Fast switch between two servo guns with a tool changer. Note: This requires the option *Servo Tool Change*.

1.3 Principles of the Spot options

Processes

The Spot functions will be controlled by separate internal program processes, which will run in parallel. For example the robot movements, the continuous supervision, and the spot welding will be handled in different independent processes. This means that if for example the program execution and thus the robot movements is stopped, then the welding and supervision will continue until they come to a well defined process stop. For example, the welding process will carry on and finish the weld and open the gun, although the program has been stopped during the weld phase.

Routines

For well defined points in the welding sequence and movements, calls to user routines offer adaptations to the plant environment. A number of predefined parameters are also available to shape the behavior of the Spot instructions.

1.4 Programming principles

Instructions

Both the robot movement and the control of the spot weld equipment are embedded in the basic spot weld instructions `SpotL` and `SpotJ`. These are used for sequential welding and are available in all spotweld options. If welding with several guns simultaneously then `SpotML` or `SpotMJ` has to be used.

Process definitions

Each spot welding process is specified by:

- `Spotdata`: spot weld process data
- `Gundata`: spot weld equipment data
- The system modules SWDEFINE and SWDEFUSR: RAPID routines and global data for customizing purposes for example adaptations for a specific process equipment.
- The system module SWUSER: RAPID routines and global data for changing of process and test behavior.
- System parameters: the I/O signal configuration and the manipulator configuration. See *Operating manual - IRC5 with FlexPendant* and *Technical reference manual - System parameters*.

1.5 Spot welding instructions

Instruction	Used to
SpotL	Control the motion, gun closure/opening and the welding process. Move the TCP along a linear path and perform a spot welding at the end position with up to four robots at the same time.
SpotJ	Control the motion, gun closure/opening and the welding process. Move the TCP along a non-linear path and perform a spot welding at the end position with up to four robots at the same time.
SpotML	Control the motion, gun closure/opening and 1 - 4 welding processes. Move the TCP along a linear path and perform spot welding with 1 - 4 gun equipments at the end position.
SpotMJ	Control the motion, gun closure/opening and 1 - 4 welding processes. Move the TCP along a non-linear path and perform spot welding with 1 - 4 gun equipments at the end position.
IndGunMove	Set the servo gun in independent mode and thereafter move the gun to a specific independent position.
IndGunMoveReset	Reset the independent mode for servo gun.
SetForce	Close the gun a predefined time then open the gun.
OpenHighLift	Open the pneumatic gun to the highlift position (large gap).
CloseHighLift	Close the pneumatic gun to the work stroke position (small gap).
CalibL	Calibrate the servo gun during linear movement to the programmed position.
CalibJ	Calibrate the servo gun during non-linear movement to the programmed position.
Calibrate	Calibrate the servo gun in current position without movement.
STTune	Tune motion parameters for the servo gun.
STTuneReset	Reset tuned motion parameters for the servo gun.
MeasureWearL	Measure the tip wear and recalculates the TCP. Only available if *Spot Servo Equalizing* is installed.
ReCalcTCP	Calculates the tip wear and recalculates the TCP. Only available if *Spot Servo Equalizing* is installed

1.6 Spot welding data

Data type	Used to define:
spotdata	The spot weld process
gundata	The spot weld equipment
forcedata	The `SetForce` process
simdata	Simulation modes

2 Programming

2.1 Spotweld programming

Introduction

This chapter describes the basic functions and steps to take when creating, testing, and running spot weld programs with Spot options.

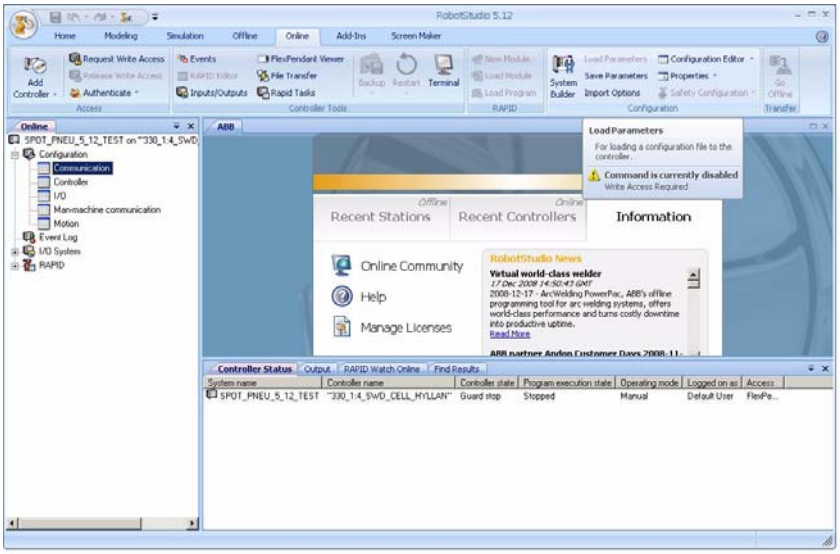
It is assumed that the servo gun is installed and force calibrated. If not see section *Motion control* on page 191, and *Application manual - Servo gun tuning*.

2.2 Quick start

Install servo gun parameters

If the system is cold started, the servo gun parameters can not be installed. Load the gun parameters from the FlexPendant using the Configuration editor. Tap **Add new parameters** and restart the system.

The servo gun parameters can also be loaded using the RobotStudio **Configuration Editor**.



Load servo gun parameters

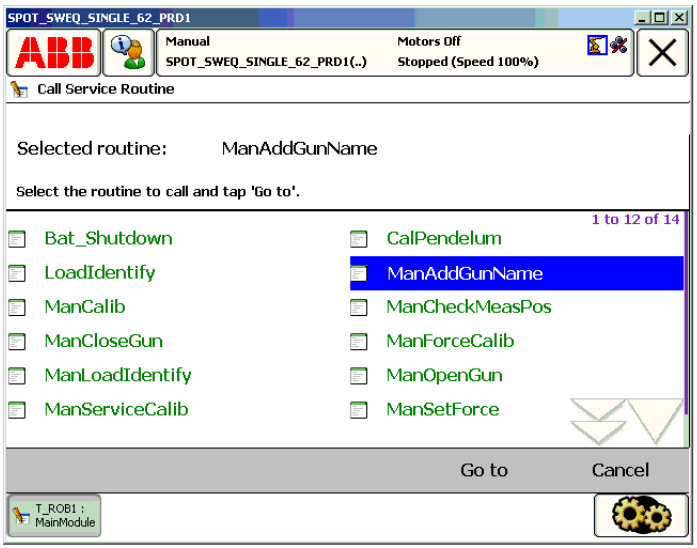
Note!

This step can be skipped if pneumatic guns are used.

Set the servo gun name

After the gun parameters are installed and the system is restarted, the gundata needs to be updated with the servo gun name (mechanical unit name) so the spot instructions will work correctly. This is done with the service routine *ManAddGunName*.

In the Program Editor, tap **Debug** and then tap **Call Service Routine**



	Action	Note
1.	Run the service routine **ManAddGunName** to find all configured servo guns in the system, and add their names to the gundata array (curr_gundata). See *gundata - Equipment specific weld data* on page 138	Follow the instructions in the routine.
2.	Ready.	

Note!

This step can be skipped if pneumatic guns are used.

Gun calibration

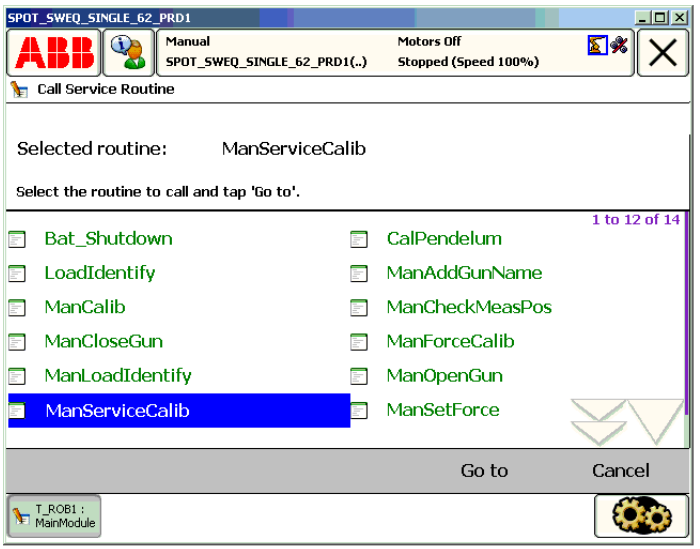
After installing the gun parameters and restarting the system, the gun must be synchronized by a fine calibration. Apart from other kinds of additional axes, this action also requires running a RAPID service routine, **ManServiceCalib** to initialize or synchronize the gun position.

It will not be possible to run any Spot instructions until the gun is synchronized and a service calibration to find the contact position for the gun has been done.

Note!

Check that a force calibration has been done for the gun before running this service routine. If the gun is **not** force calibrated and properly tuned, follow the tuning procedure described in the *Application manual - Servo gun tuning.*

In the Program Editor, tap **Debug** and then tap **Call Service Routine**



Service calibration

Fine calibrating

Fine calibration must be performed when installing a new servo gun or if the servo gun axis is in state *Not Calibrated*.

	Action	Note
1.	Fine calibrate the axis using the Calibration window. The axis can be in any position.	
2.	Run the service routine **ManServiceCalib** option 2.	If the gun is considered to be force calibrated (force_calib = true) the gun will move to zero position and close slowly until tip contact is detected. Otherwise a warning will be displayed with a question whether the gun has been force calibrated or not. Answering **Yes** will perform the service calibration anyway, **No** will end the routine. **NOTE!** If the gun is **not** force calibrated and properly tuned, follow the tuning procedure described in the *Application manual - Servo gun tuning.*
3.	As a result, the gun position is updated to be zero in the position of contact and the tip wear value is reset.	
4.	Ready.	

Warning!

If the boolean **force_calib** is set to **TRUE** without having performed the force calibration procedure, the servogun can be damaged, please make sure that the servo gun is force calibrated and tuned and that the force calibration values are correct before setting this parameter, see *Application manual - Servo gun tuning*.

2.3 The spotweld instructions for sequential welding

`SpotL` and `SpotJ` are the basic spotweld instructions in the Spot options. The instructions includes a movement to the weld position and performing the desired weld process. They contains basically the same type of information as a positioning instruction, but also arguments that serve as data for the spotweld process. These instructions are used for welding with one gun or welding with several guns in sequence.

For further details, see section *SpotL/SpotJ - The basic spot welding instructions* on page 73.

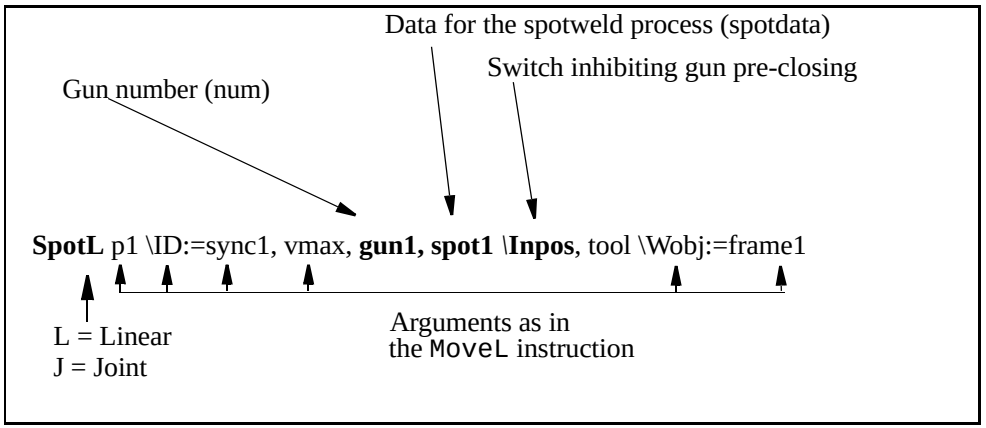


Figure 1

2.4 Defining spotweld data

Before starting to program the instructions, define the spotweld data to be used. This data is divided into two types:

- `spotdata`; describes the spotweld process specific data for a specific spot. See *spotdata - Spot weld data* on page 148.
- `gundata`; describes spotweld gun characteristics and other equipment specific data. All gun equipments used are defined in the gundata array curr_gundata in SWUSER. Change the default setup so it corresponds to the gun equipments used. See *gundata - Equipment specific weld data* on page 138.

2.5 Programming spotweld instructions

	Action	Note
1.	Jog the robot to the desired destination position and jog also the gun axis to desired preclose tip position	Servoguns only
2.	In the Program Editor, tap **Add instruction**, and then select **Motion&Proc** from the list of instructions.	
3.	Select the instruction **SpotL** or **SpotJ**	The instruction will be added directly to the program. The arguments are set in relation to the last programmed spot-weld instruction.
4.	Change the arguments if needed.	
5.	Add more spot weld instructions the same way.	

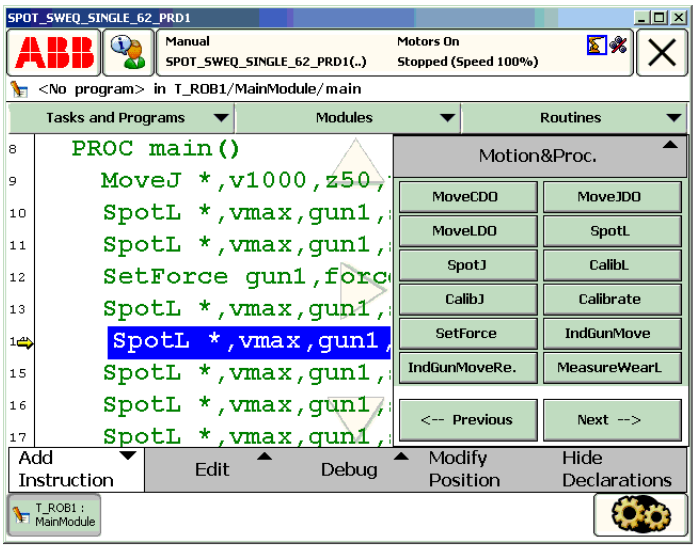


Figure 2 Programming spotweld instructions

2.6 Programming example 1 for a servo gun

In this example a single servo gun (gun1) is used, held by the robot. Four spots are to be welded with two different `spotdata` used, *spot10* and *spot20*. The data is created in advance. Current gun parameters are set in the first `gundata` in the *curr_gundata* array in SWUSER.

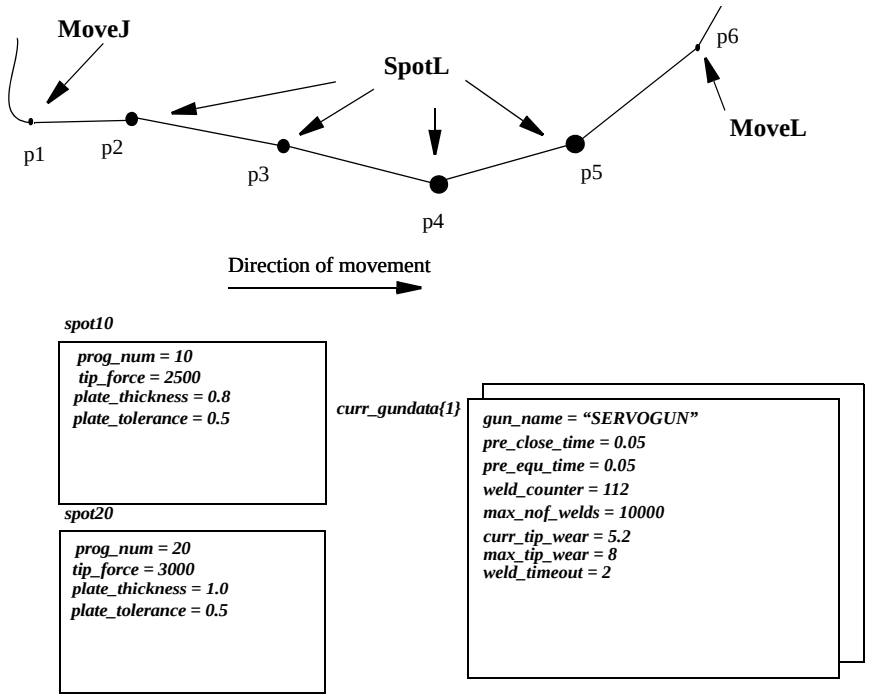


Figure 3 Programming example for Spot welding with a servogun.

RAPID code sequence:

```
MoveJ p1, v600, z50, tool1;
SpotL p2, vmax, gun1, spot10, tool1;
SpotL p3, vmax, gun1, spot10, tool1;
SpotL p4, vmax, gun1, spot20, tool1;
SpotL p5, vmax, gun1, spot20, tool1;
MoveL p6, v600, z50, tool1;
```

2.7 Programming example 2 for a pneumatic gun

In this example a single pneumantic gun (gun1) is used, held by the robot. Four spots are to be welded with two different `spotdata` used, *spot10* and *spot20*. The data is created in advance. Current gun parameters are set up in the first `gundata` in the *curr_gundata* array in SWUSER.

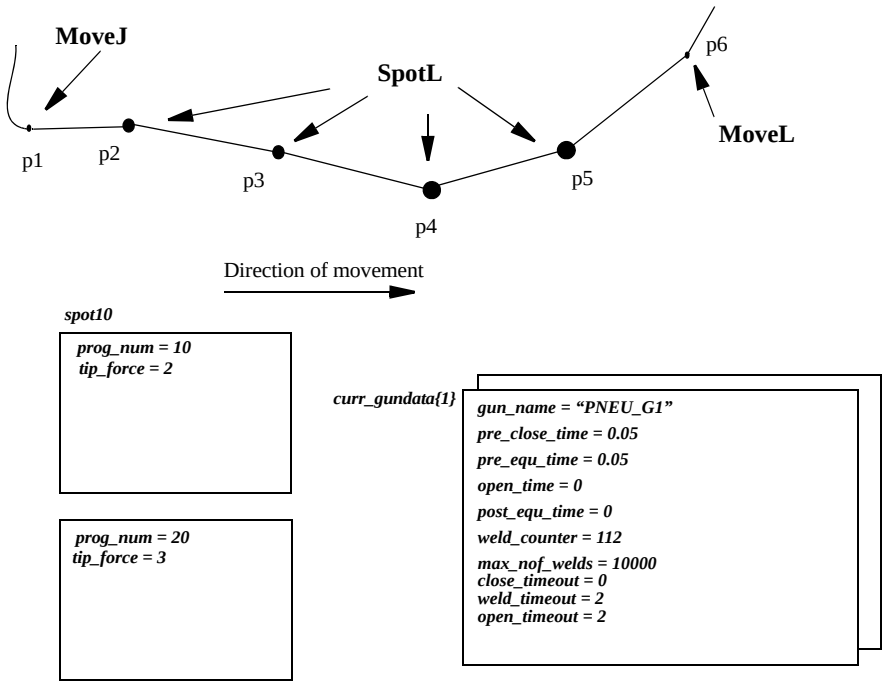


Figure 4 Programming example for Spot welding with a pneumatic gun.

RAPID code sequence:

```
MoveJ p1, v600, z50, tool1;
SpotL p2, vmax, gun1, spot10, tool1;
SpotL p3, vmax, gun1, spot10, tool1;
SpotL p4, vmax, gun1, spot20, tool1;
SpotL p5, vmax, gun1, spot20, tool1;
MoveL p6, v600, z50, tool1;
```

2.8 Editing spotweld instructions

Changing the current spotdata

	Action	Note
1.	Select current `spotdata` in the instruction.	
2.	Tap **Edit**, and then tap **Change Selected**.	
3.	Change the value.	
4.	Tap **OK**.	

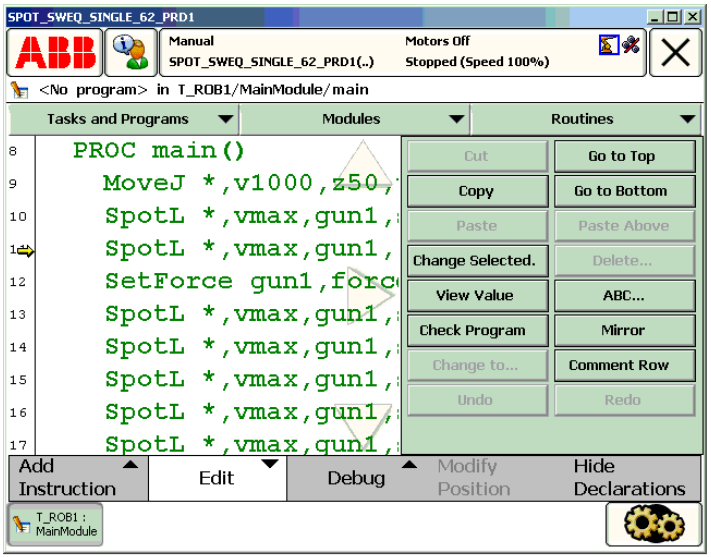


Figure 5 Editing spotweld instructions

Changing to another spotdata

	Action	Note
1.	Select current `spotdata` in the instruction.	
2.	Tap **Enter**.	
3.	Select the `spotdata` from the list.	
4.	Tap **OK**.	

2.9 Testing spotweld instructions with simulated welding

To prevent the spotweld process executing during programming, it is possible to run the program in simulation mode.

This can be done by setting `sim_type = 2` in *curr_simdata* in SWUSER. This will set the output *enable_current* low and the simulation will be carried out by the weld timer.

If such a signal not is connected the spotweld can be internally simulated by setting `sim_type = 1` in *curr_simdata*. The simulated weld time used is the time `sim_time` in *curr_simdata*. In this simulation mode the start signal is never sent to the welding timer.

When simulation is active it is also possible to run without closing the gun or without testing plate thickness (servo guns only). This is done by setting `inhib_close` or `no_plates` to `TRUE` in *curr_simdata*.

If *Spot Servo Equalizing* is installed it is possible to set `sim_type = 3` to activate the weld position Touch Up function. See *Software Equalizing* on page 41.

2.10 Jogging the robot after unintentional gun disconnection

If the motor cables are unintentional disconnected when the servo gun is activated, the servo gun must be deactivated in order to jog the robot to a service position. Deactivation is done in the **Jogging** window by selecting axis and tapping **Deactivate**. After service or repair the revolution counter must be updated since the position has been lost.

Note!

See *Motion control* on page 191.

2.11 Gun preclosing

The spotweld instructions have a built-in preclosing of the weld guns, that is when approaching the position the guns will start to close in advance to save time.

The gun closing time, `pre_close_time`, has to be defined for each used gun in the `gundata` array *curr_gundata* in SWUSER.

For servo guns the gun closure is coordinated internally which means that the gun is not closed to the plates before the robot is in position.

The data `pre_close_time` is not used if *Software Equalizing* is active, see *Software Equalizing* on page 41.

Note!

The preclosing can be disabled by using the `\InPos` argument in the instruction.

2.12 Activating mechanical equalizing systems

The spotweld instructions have a function for equalizing mechanical equalizing systems in the gun, that is when approaching the position a signal is activated to be used for the equalizing. The signal is deactivated after the weld process before the next robot motion is released.

The gun equalizing time, `pre_equ_time`, is defined for each used gun in the `gundata` array *curr_gundata* in SWUSER.

2.13 Tip dressing for servo guns

The `gundata` contains counters and tip wear information for each used gun. The counters will be automatically incremented for each spot and the tip wear information is updated after each gun calibration. This information can be used to decide when to do next tip dressing or tip exchange.

2.14 Manual actions

Some useful service routines are predefined to be used for manual actions during programming and test.

- Tap **Debug**, and then tap **Call Service Routine**

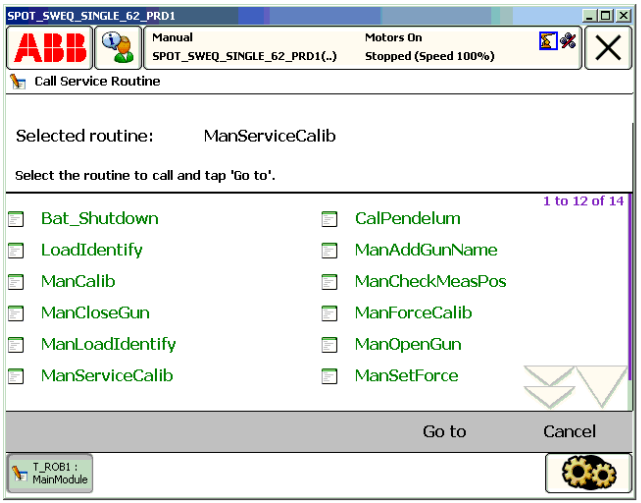


Figure 6 Manual actions

The following service routines are predefined for manual actions:

ManAddGunName	Searches the system for available servoguns and add their names to gun name in current `gundata`.
ManCloseGun	Close the gun according to data in `man_forcedata`. The gun equalize signal is also activated.
ManOpenGun	Open the gun. The gun equalize signal is deactivated.
ManOpenHighLift	Open the pneumatic gun to the large stroke position.
ManCloseHighLift	Close the pneumatic gun to the work stroke position.
ManSpot	Perform a weld in current position according to data in *curr_spotdata*.
ManSetForce	Perform a SetForce action according to data in `man_forcedata`. The gun equalize signal is also activated/ deactivated.
ManCalib	Perform a calibration of the servo gun.
ManForceCalib	Perform a force calibration of the servo gun.

ManServiceCalib	Option 1: Synchronize the servo gun without jogging after the revolution counter has been updated, the servo gun will close slowly until it reaches the contact position. Option 2: Synchronize the servo gun without jogging after the gun has been fine calibrated, the servo gun will move to zero position and then close slowly until it reaches the contact position.
ManCheckMeasPos	To verify if a selected position is good enough for the tip wear measuring function. Only available if *Spot Servo Equalizing* is installed.

If several guns are used then a dialog will appear asking for the gun number of the gun to be handled.

2.15 Spotweld instructions for simultaneous welding with multiple guns

`SpotML` and `SpotMJ` has to be used if welding with several guns at the same time is desired. It is possible to use four guns simultaneously. The instruction includes a movement to the weld position and performing the desired weld processes. It contains basically the same type of information as a positioning instruction but also arguments that serve as data for the different spotweld processes.

See *SpotML/SpotMJ - Spot welding with multiple guns* on page 89.

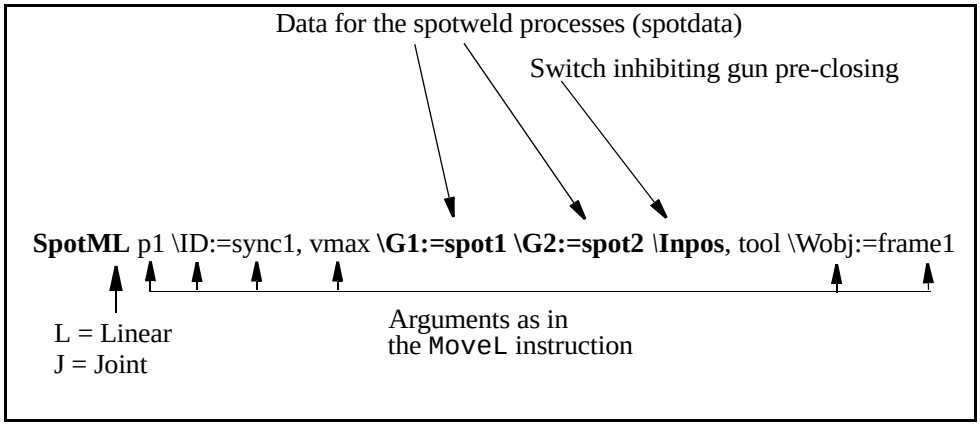


Figure 7 Data for the spotweld processes

2.16 Programming example 3 for two servo guns

In this example two different stationary guns are used, mounted close to each other. The robot is holding the work piece. Seven spots are to be welded with two different `spotdata` used, *spot10* and *spot20*. Current gun parameters has been set up in the first and second `gundata` in the *curr_gundata* array in SWUSER.

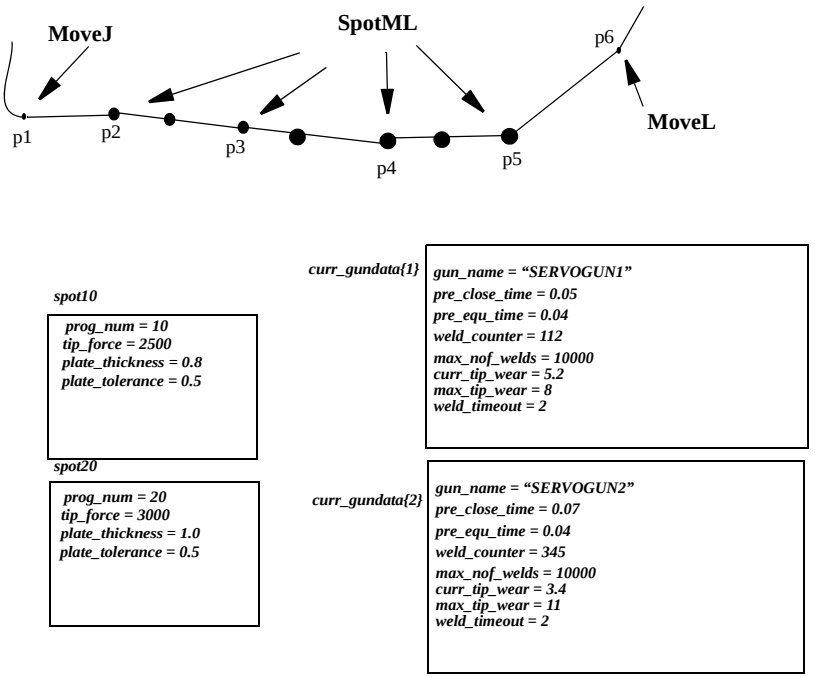


Figure 8 Programming example for two servo guns.

RAPID code sequence:

```
MoveJ p1, v600, z50, multi_gun\Wobj:= frame1;
SpotML p2, vmax\G1:=spot10\G2:=spot10,multi_gun\Wobj:=
   frame1;
SpotML p3, vmax\G1:=spot10\G2:=spot20, multi_gun\Wobj:=
   frame1;
SpotML p4, vmax\G1:=spot20\G2:=spot20, multi_gun\Wobj:=
   frame1;
SpotML p5, vmax\G1:=spot20, multi_gun\Wobj:= frame1;
MoveL p6, v600, z50, multi_gun\Wobj:= frame1;
```

2.17 Programming example 4 for two pneumatic guns

In this example two different stationary guns are used, mounted close to each other. The robot is holding the work piece. Seven spots are to be welded with two different `spotdata` used, *spot10* and *spot20*. Current gun parameters has been set up in the first and second `gundata` in the *curr_gundata* array in SWUSER.

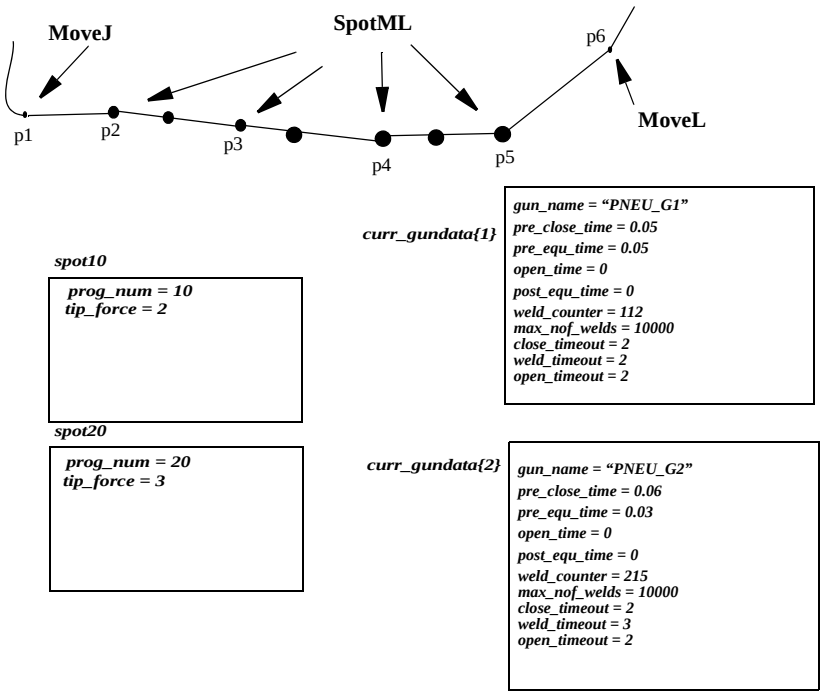


Figure 9 Programming example for two pneumatic guns.

RAPID code sequence:

```
MoveJ p1, v600, z50, multi_gun\Wobj:= frame1;
SpotML p2, vmax\G1:=spot10\G2:=spot10,multi_gun\Wobj:=
   frame1;
SpotML p3, vmax\G1:=spot10\G2:=spot20, multi_gun\Wobj:=
   frame1;
SpotML p4, vmax\G1:=spot20\G2:=spot20, multi_gun\Wobj:=
   frame1;
SpotML p5, vmax\G1:=spot20, multi_gun\Wobj:= frame1;
MoveL p6, v600, z50, multi_gun\Wobj:= frame1;
```

2.18 Weld process timing for servo guns

The graphic in *Weld process timing for servo guns* on page 35 shows the weld process timing for a servo gun and where in the sequence the user hooks will affect the internal behavior.

If welding is done with several guns at the same time then each process is handled in separate tasks independent of each other.

The system parameter `Post_sync_time` (*Post-synchronization Time*) in the topic *Motion*, type *SG Process*, defines the predicted release time of the next robot movement after a weld. Can be used to shorten the cycle time.

The robot will start to move before the gun is completely opened. Default value is 0.

If *Soft Equalizing* is activated the `pre_equ_time` is not used. The preclosing of the gun is in this case handled automatically, see *Software Equalizing* on page 41.

Note!

The value of this parameter (*Post-synchronization Time*) can affect the cycle time of the program negatively if for example two welding points are programmed at the same position. To minimize this risk the value can be increased. See *Application manual - Additional axes and stand alone controller.*

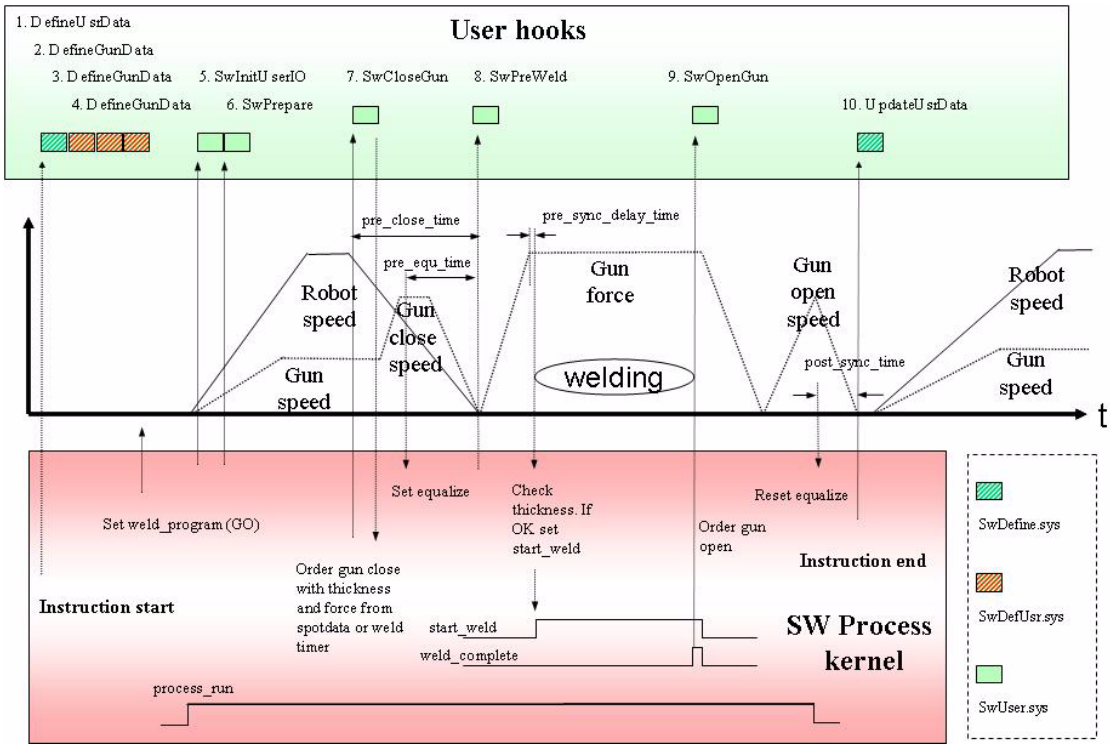


Figure 10 Weld process timing for servo guns

2.19 Weld process timing for pneumatic guns

The following graphic shows the weld process timing for a pneumatic gun and where in the sequence the user hooks will affect the internal behavior.

If welding is done with several guns at the same time then each process is handled in separate tasks independent of each other.

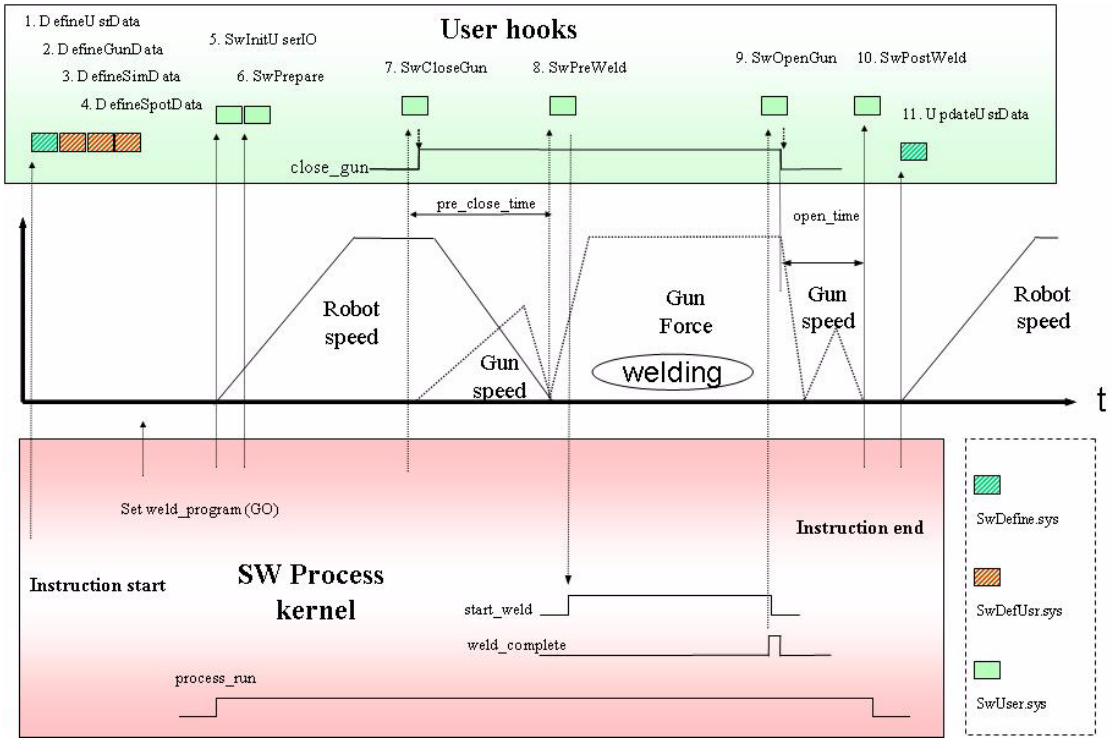


Figure 11 Weld process timing for pneumatic guns

2.20 Weld process timing for software equalizing with servo guns

The following graphic shows the weld process timing for a servo gun and when software equalizing is activated, and where in the sequence the user hooks will affect the internal behavior.

If welding is done with several guns at the same times then each process is handled in separate tasks independent of each other.

The system parameter `Post_sync_time` (*Post-synchronization Time*) in the topic *Motion*, type *SG Process*, defines the predicted release time of the next robot movement after a weld. Can be used to shorten the cycle time.

The robot will start to move before the gun is completely opened. Default value is 0.

Note!

If *Soft Equalizing* is activated the `gundata` parameters `pre_close_time` and `pre_equ_time` is not used. The preclosing of the gun is in this case handled automatically, see *Software Equalizing* on page 41.

Note!

The value of this parameter (*Post-synchronization Time*) can affect the cycle time of the program negatively if for example two welding points are programmed at the same position. To minimize this risk the value can be increased. See *Application manual - Additional axes and stand alone controller.*

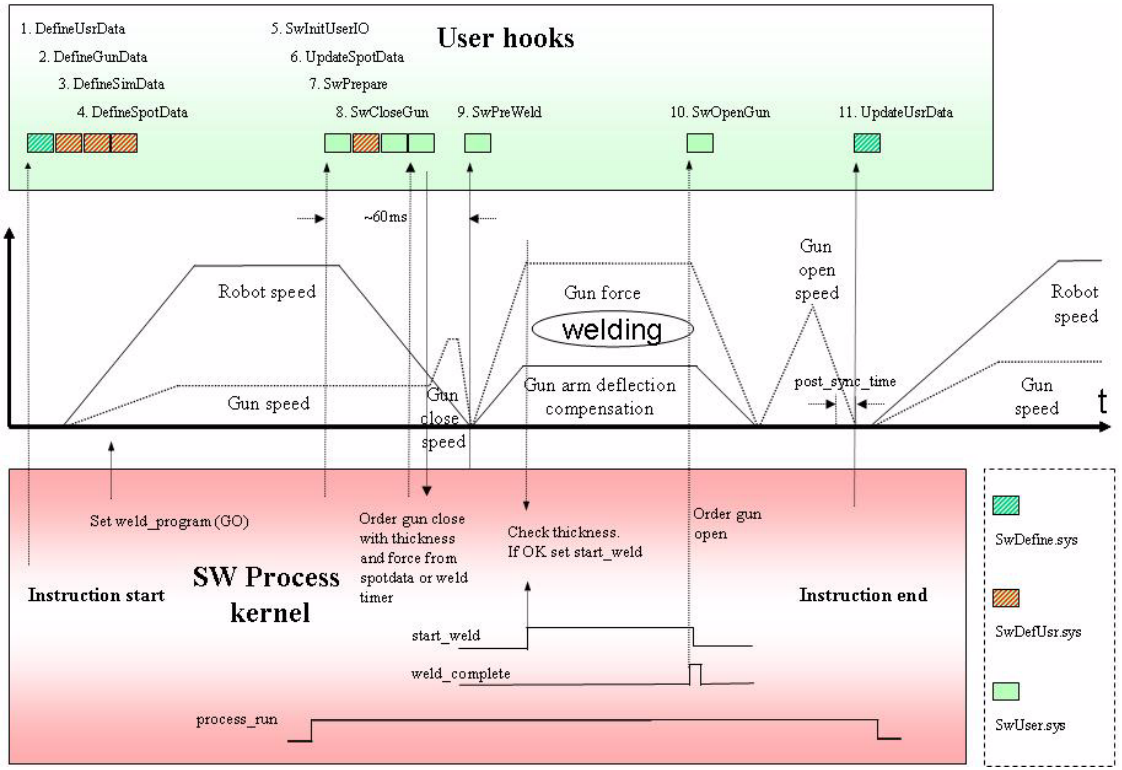


Figure 12 Weld process timing if software equalizing is activated.

2.21 Pneumatic spot welding gun and gripper

When the robot has a pneumatic spot welding gun and a gripper, with or without a tool changer, it takes some special arrangements to control the clamps on the gripper. The reason is that the air pressure valve on the Media Panel is controlled by the weld timer, which uses the valve to obtain different gun forces. The weld timer is in control of the air pressure valve, even when the robot is holding the gripper, so the following preparations have to be done to control the clamps on the gripper:

- In the weld timer, create weld programs for the desired pressures for gripper control.
  Note! Weld current MUST be deactivated in the programs.
- In the RAPID code, create the necessary control routines, and include `SetGO` instructions that sets the group output to the program number to the corresponding program in the weld timer.
  Note! The *gX_new_prog* signal must be on at all times for the air pressure valve to follow immediately a new program number.

3 Software Equalizing

3.1 Introduction to SoftWare Equalizing

Introduction

This chapter describes the Software Equalizing functions, that is functions that make it possible to use spot welding guns without mechanical equalizing systems. These functions are available if *Spot Servo Equalizing* is installed.

Recommendations

When guns without mechanical equalizing systems are used it is **very** important to have good accuracy when the TCP is defined and when the weld positions are teached.

It is also important to handle the tip wear and recalculate the TCP regularly and also to release the fixed gun arm from the sheet when the gun is moved between weld positions. For most guns it is also necessary to handle the gun arm deflection.

Available functions

The Software Equalizing functions are a number of functions intended to handle these issues for the user. However, it is not always necessary to use all functions. It depends on desired accuracy, sheet stiffness, gun properties as type, size and stiffness and so on.

The following Equalizing functions are available:

- Weld position Touch Up.
- Release of the fixed gun arm.
- Gun arm Deflection Compensation.
- Tip Wear Measurement and Compensation.

It is possible to use guns with mechanical equalizing and guns using Software Equalizing in the same user program. The equalizing type is determined by a data in the `gundata` for each used gun, *softw_equ*. For more information, see *gundata - Equipment specific weld data* on page 138.

3.2 Some basic definitions

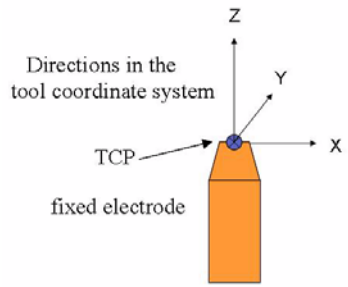
How to define the TCP

The TCP has to be defined, as normally, on the tip of the fixed gun arm.

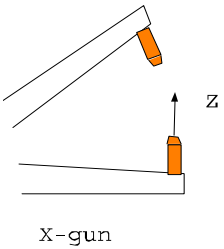
It is important to define also the z-direction in the tool coordinate system since all automatically search movements and compensations will be done in the z-direction. See the graphics below.

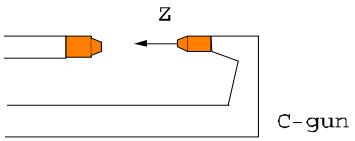
Normally the z-direction should point out from the fixed tip but it is also possible to have the z-coordinate defined in the opposite direction, but then the setup data *opposite_z* has to be set to `true` in the user module SWDEFUSR, see *Setup data* on page 168.

See *Setup data for Software Equalizing* on page 58.



The z-direction when different gun types are used:



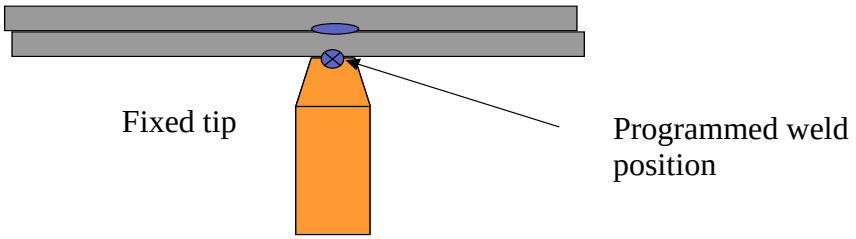


How to setup the tool TCP

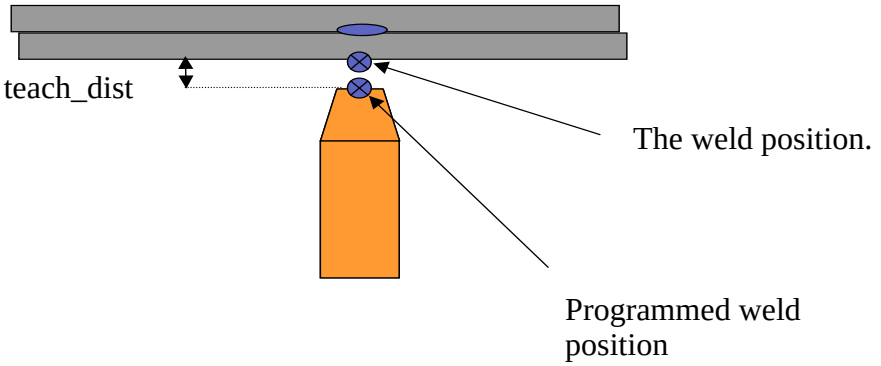
	Action	Note
1.	Define the TCP for the used gun with a new tip mounted. Use the 5 point method.	See *Operating manual - IRC5 with FlexPendant*
2.	Store the result in a `tooldata` for example *ref_tool1*	
3.	Save the tip (tool) as a reference tip (the physical tip)	

How to program the weld positions

.



The weld position is normally teached in the position where the fixed electrode is touching the sheet during the weld process. See the graphic above.



It is also possible to program the weld positions a selected distance from the sheet. Programming like this is a commonly used method when spot weld guns with mechanical equalizing systems are used. Some users also think it is easier to get a good accuracy programming with a short distance to the sheet.

If this method is selected, and described for the programmers, the selected distance from the tip to the sheet during programming has to be set in the setup data `teach_dist` in the user module SWDEFUSR, see *Setup data* on page 168.

For more information, see *Setup data for Software Equalizing* on page 58. Note, this programmed weld position is only used during programming and touch up. During program execution the target position is of course always the weld position on the sheet.

However, the default behavior is programming without teach distance, that is. `teach_dist = 0`.

Note!

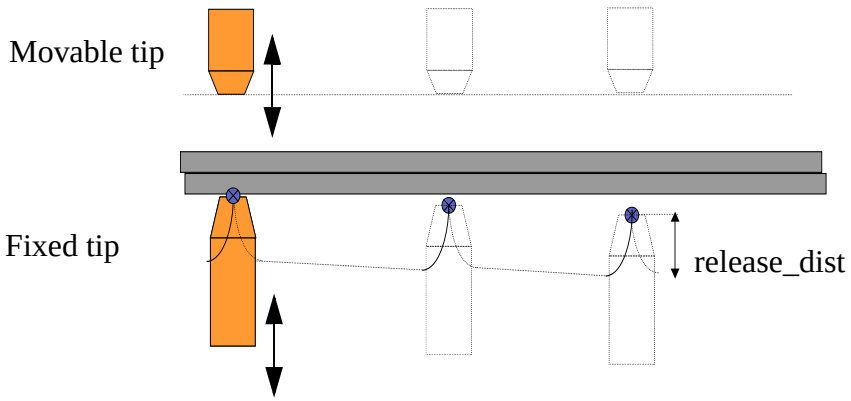
Before touching up the weld positions a `MeasureWearL` with the `Reference` switch has to be done.

3.3 Weld position Touch Up

Introduction

Weld position Touch Up is a support function used in manual mode to get a faster and easier way to adjust the programmed weld positions. During the touch up it is possible to change the fixed gun tip in the z-direction and it is also possible to change the position for the movable gun arm.

Normally a touch up has to be done at least once in the beginning after manual or offline programming of the weld positions.



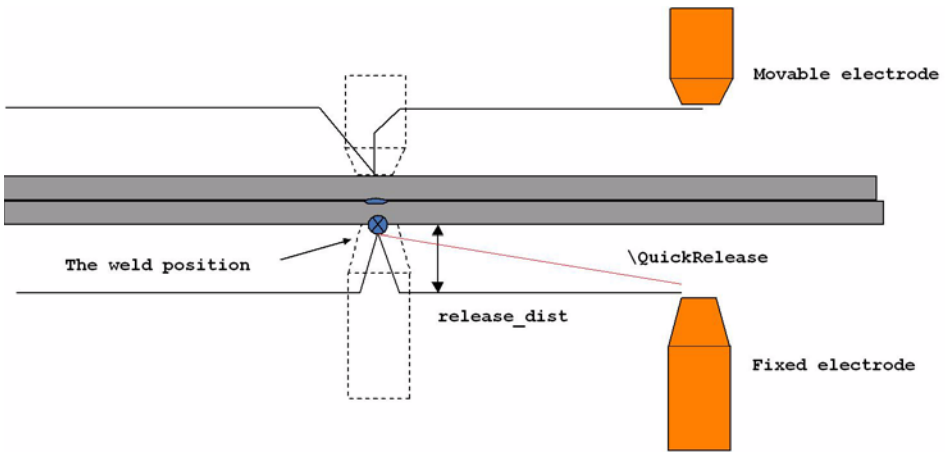
How to use the weld position touch up function

	Action	Note
1.	Make sure that current TCP is relevant for the used tip, define the tool data correctly.	
2.	Set the system in weld position touch up mode (`simtype = 3` in *curr_simdata*)	See *simdata - Simulation data* on page 145.
3.	Start the program. The robot is running the program as normal, but all `SpotL/J` instructions are executed as move instructions	

	Action	Note
4.	The robot stops in each programmed weld position and a user interaction is started which gives possibilities to confirm and directly go to next spot or adjust current position	
5.	During adjustment the fixed gun tip is moved in small steps to the sheet or to desired distance from the sheet if `teach_dist` greater than zero	See SWDEFUSR *Setup data* on page 168.
6.	It is also possible to adjust the position of the movable gun arm in a similar way	
7.	When both tips are in desired position it is possible to do a Modpos, with confirmation, to definitely reprogram the position	
8.	When the gun is moved between programmed weld positions the fixed gun tip is automatically released from the sheet. Desired distance to the sheet is a user defined data predefined for each used gun, `release_dist`. This release movement can be skipped to save cycle time if the optional switch `\QuickRelease` is selected in the `SpotL/J` instruction.	See *gundata - Equipment specific weld data* on page 138. See *SpotL/SpotJ - The basic spot welding instructions* on page 73.
9.	When all weld positions are checked, set `simtype = 0` to disconnect the Touch Up mode	See *gundata - Equipment specific weld data* on page 138.

3.4 Release of the fixed gun arm

This function is used to get an automatic release of the fixed gun arm from the sheet, when the gun is moved between the weld positions during normal program execution.



During execution of `SpotL/J` instructions, the robot moves the gun to the weld position, via a position a release distance from the sheet.

After the weld when the gun is opened, an extra movement is performed to release the fixed gun arm from the sheet except if the `QuickRelease` functionality is activated in the `SpotL/J` instruction.

If the `\QuickRelease` switch is selected in the `SpotL/J` instruction the release distance movement after the weld will be skipped, this may save some cycle time and can be used when the spots are located close toghether.

Note!

The `\QuickRelease` function is suitable when programming close weld positions, not when there are large distances between the weld positions.

The release distance, `release_dist`, is a user defined data predefined for each used gun. see *gundata - Equipment specific weld data* on page 138.

This function is disabled if `release_dist = 0` or `softw_equ = FALSE` in current gundata.

Note!

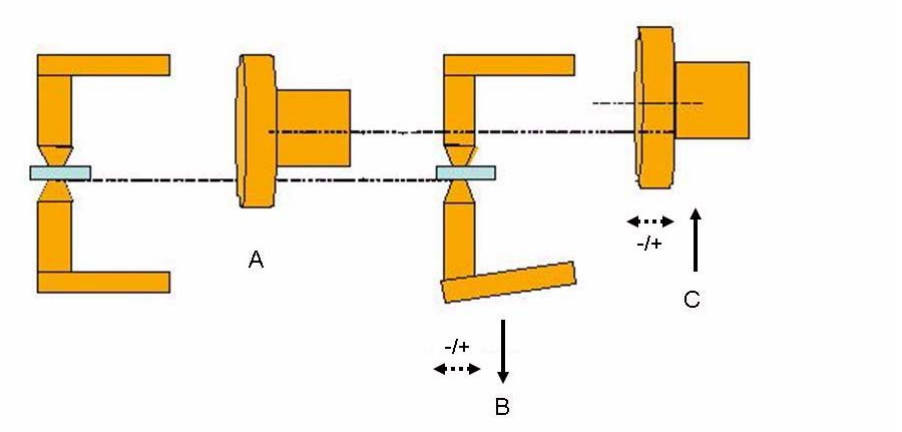
To get a good synchronization between the release movement and the gun opening when software equalizing is used, the gun is always held in the closed state 40 ms extra after weld complete. To save cycle time it is therefor possible to reduce the programmed hold time in the weld timer this amount of time (2 periods).

3.5 Gun arm deflection compensation

Introduction

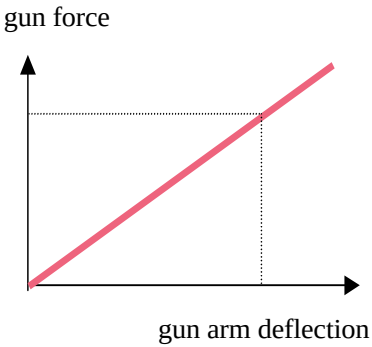
Normally there is a need to compensate for gun arm bending in the tool z-direction, but it is possible to configure gun arm bending compensation in other directions as well, and the deflection can also be done in a opposite direction if needed by specifying a negative deflection value in gundata, see *Customizing* on page 237.

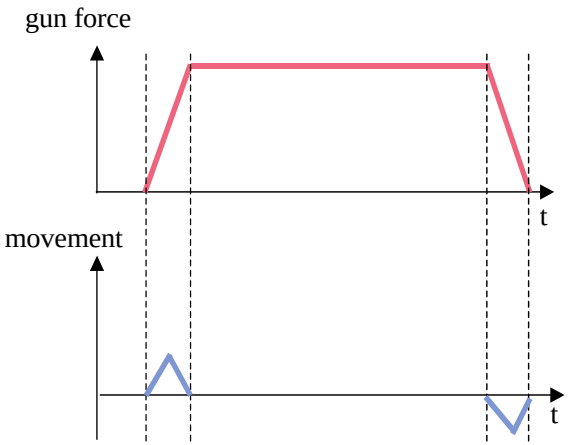
With this function the gun arm deflection is automatically compensated with an extra robot movement when the gun is closed and the gun force is applied. See graphic below.



A	Robot flange
B	Gun arm deflection
C	Robot compensation movement

The dashed arrows symbolizes deflection compensation in the tool x-direction.





Data

Data for the correlation between the gun force and the arm deflection is user defined data, predefined for each used gun: `deflection_dist`, `deflection_force` and `deflection_time`, see *gundata - Equipment specific weld data* on page 138. This data is normally found in the data sheet for the used gun.

Then, during program execution of `SpotL/J` instructions, there is an added robot movement, activated at the same time as the gun force is established, to compensate for the gun arm deflection, see graphics above.

The actual gun arm deflection is calculated from the force value (`tip_force`) in current `spotdata`. Deflection calculation in `SpotL/J` instruction.

```
deflection := spotdata.tip_force * gundata.deflection_dist
  / gundata.deflection_force;
```

A movement in the opposite direction is performed after the weld, when the gun is opened. This movement is combined with the release movement.

This function is disabled if `deflection_dist = 0` or if `softw_equ = FALSE` in current `gundata`.

Tip!

It is also possible to configure gun arm deflection compensation in other tool directions, for example in the tool x-direction. See *Customizing* on page 237.

How to setup the data for gun arm deflection

.

	Action	Note
1.	Find out how much gun arm deflection there is at a specific force when the gun arms are closed, this data is normally found in the data sheet for the used gun. **TIP!** If this information is missing the gun arm deflection can be measured manually by closing the gun at a specific force, for example 4000N and measure how much the gun arm deflects related to a fixed reference position.	
2.	Enter the measured values in current `gundata`, `deflection_dist` and `deflection_force`, for example 5mm and 4000N	See *gundata - Equipment specific weld data* on page 138.
3.	Save the user module swuser.sys since the current `gundata` is located there.	

3.6 Tip wear compensation

Introduction

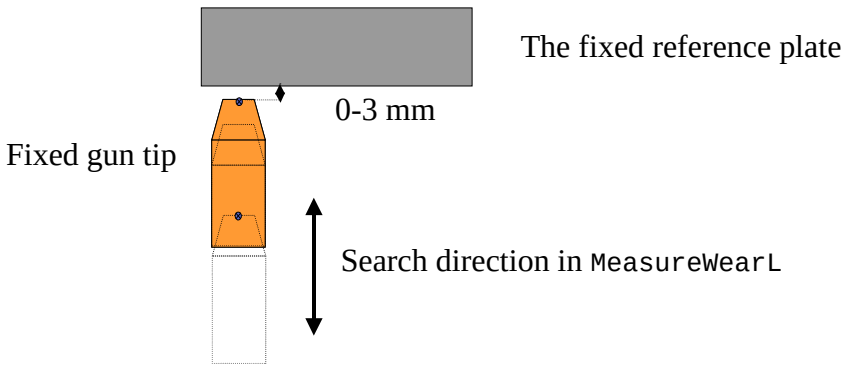
When guns without mechanical equalizing systems are used, it is important to handle the tip wear, especially the tip wear on the fixed tip, since this tip is controlling the weld position. Therefore the tip wear has to be regularly compensated during production. There are two methods available for the compensation.

Method 1: MeasureWearL, the tip wear for the fixed tip is measured and the tip wear is then compensated in current used `tooldata`.

Method 2: ReCalcTcp, the tip wear of the fixed tip is calculated based on stored information about the total tip wear and the expected relation between the tip wear of the fixed tip and the total tip wear. The tip wear is then compensated in current used `tooldata`. This method can be used as an alternative if method 1 is not suitable.

Method 1: Tip wear measurement and compensation with MeasureWearL

A RAPID instruction is available for tip wear measuring and TCP adjustment: `MeasureWearL`. This instruction is used **one** time for a reference measurement with new tips and then one time after each tip dressing and after the tips have been exchanged.



A code sequence with the measuring instruction included has to be prepared in the user program. The position in the `MeasureWearL` instruction has to be programmed close to a fix reference plate, see figure above. Also see *Measurement preparation* on page 124.

When the instruction is executed the robot first moves the gun to a start position for the search movement about 10 mm from the reference plate. Then the gun is moved until the fixed gun

tip touches the reference plate. The tip wear of the fixed gun tip is calculated. Currently used TCP value is automatically recalculated and tip wear data (`curr_wear_fix`) in current `gundata` is automatically updated.

Note!

This movement of 10mm has to take care of the wear of the tips that could be for example 20mm, it means that the total gun opening should take care of the 10mm position, the plate thickness and the total wear of the tips.

Normally when servo guns are used, gun calibration has to be done after tip dressing and after tip change. See *Calibrate - Calibrate the gun* on page 113 and *CalibL/CalibJ - Calibrate the gun during movement'* on page 108. This is necessary also when Software Equalizing is used. Preferably this calibration can be done directly after the tip wear measurement.

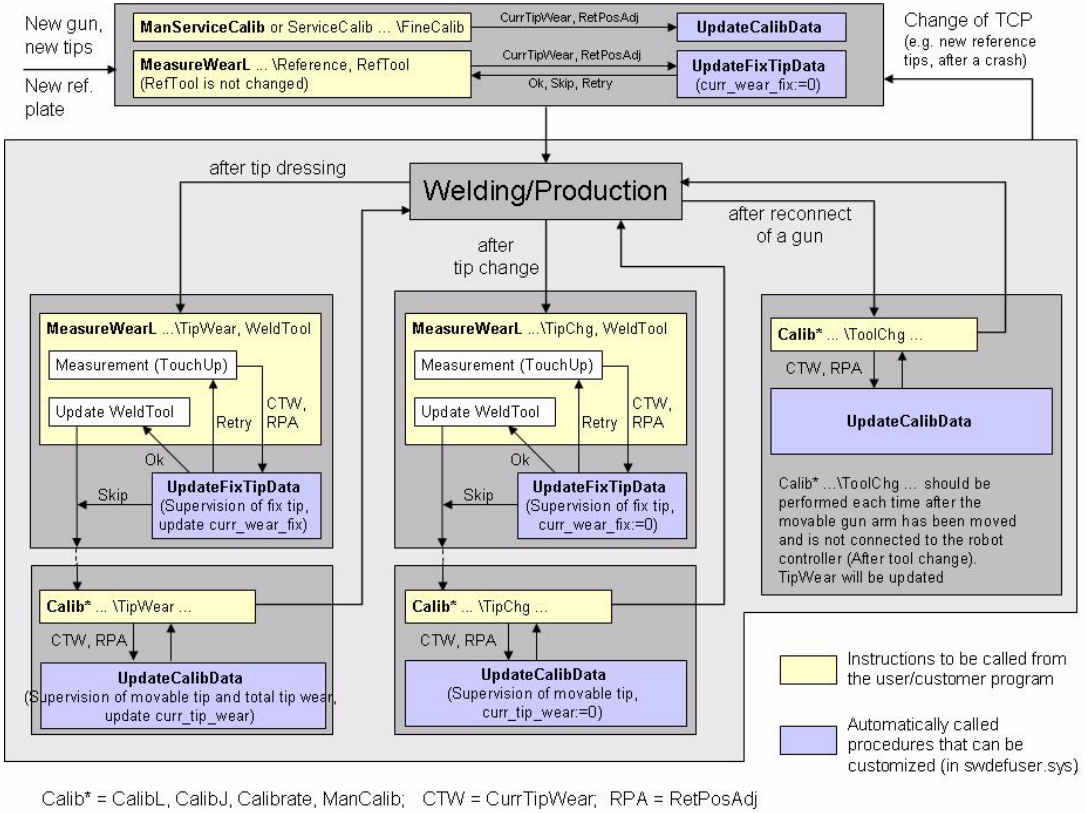
The reference plate can be mounted in an optional position in the work range, preferably on the tip dresser stand. But it is necessary to orient the tool in the measuring position in that way that an additional torque is generated on at least one of the robot motors when the robot is touching the fixed plate, preferably **axis 4 to 6**.

It is possible to verify if the selected measuring position or gun orientation is good enough by using a service routine; *ManCheckMeasPos*. Just run the service routine when the robot is in the selected position and you will get status information on the display.

How to find a good measuring position

	Action	Note
1.	Jog the robot to the position where the fixed plate is mounted, for example on the tip dresser stand.	
2.	Run the service routine *ManCheckMeasPos* to find out if the position is good or not, if not jog/reorient the robot to a new position.	See *Service routines (Manual Actions)* on page 164.
3.	Create a program with a code sequence with the measuring instruction included.	See *MeasureWearL - Measure current electrode wear* on page 123.

Tip measurement sequence



Calib* = CalibL, CalibJ, Calibrate, ManCalib; CTW = CurrTipWear; RPA = RetPosAdj

The reference measurement with `MeasureWearL\Reference` will calibrate the position of the reference plate for the tip wear measurement with the robot. The parameters of the tool used for the reference measurement (RefTool) and the calibration values are stored in the persistent variables *tw_ref_tool* and *tw_ref_dist* located in the user module swuser.sys. All following calls of `MeasureWearL` (`\TipWear`, `\TipChg`) measures only the difference to the reference position.

The tips (or the real TCP) and the tool (RefTool) used for the reference measurement must be the same as used for teaching of the weld positions (robtargets).

The reference measurement needs to be done again when the reference plate has been moved or the TCP has been changed (for example after a crash) or the `\RefChange` switch can be used instead, see *Measurement after reference plate changed* on page 126.

For the measurements the robot contacts the reference surface always with the same force (setup data `m_touch_force` in the user module swdefusr.sys).

Since the calibration values are stored in swuser.sys the file should be saved after the reference measurement.

Input values of *CurrTipWear* and *RetPosAdj* in *UpdateCalibData* and *UpdateFixTipData*.

	CurrTipWear	RetPosAdj
Calib* ... \TipWear	0	Difference to the last calibration of the gun. Normally last call with \TipWear. Total difference between the new and the old (worn) tips.
Calib* ... \TipChg	Total wear of both tips	Difference to the last calibration of the gun.
Calib* ... \ToolChg	Total wear of both tips	Difference to the last calibration of the gun.
MeasureWearL .. \TipChg	0	Difference between the actual and the tip used for the reference measurement.
MeasureWearL .. \Reference	0	Measured reference distance.

See *MeasureWearL - Measure current electrode wear* on page 123.

NOTE!

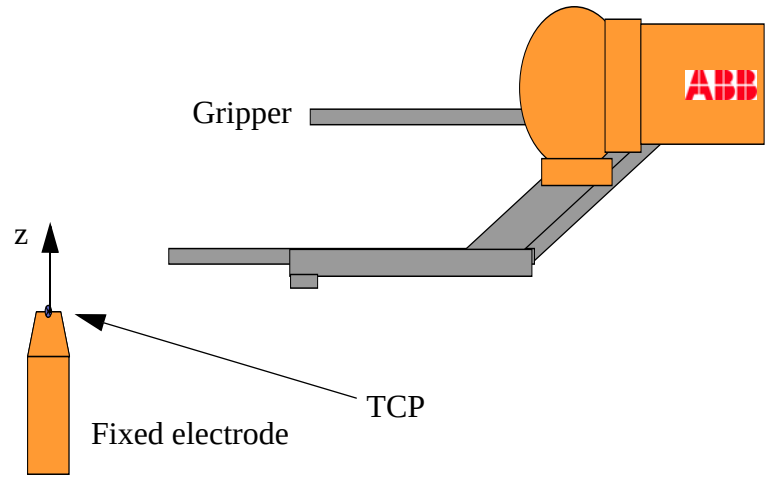
For some special configurations the `MeasureWearL` is less suitable, for example very large guns and/or when an acceptable touch up position is not possible to reach for some reason. Then the `ReCalcTcp` method should be used instead.

Tip wear measurement and compensation for stationary guns

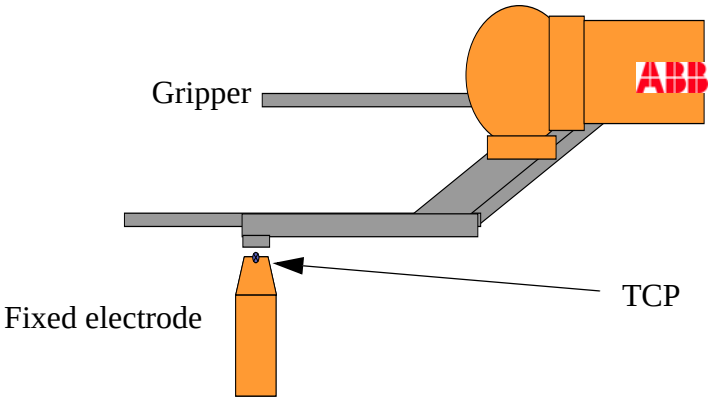
Generally when stationary tools are used, the work object is held by the robot and related to the wrist coordinate system and the TCP, still defined on the tool, is related to the world coordinate system. This is the case also when stationary guns are used. As before, the TCP has to be defined on the fixed tip and the z-direction has to be defined, see figure below.

The parameter `robhold` in current `tooldata` and `wobjdata` defines whether the robot is holding the gun or not.

For more information about how to define the stationary tool coordinate system, see *Operating manual - IRC5 with FlexPendant.*



All Software Equalizing functions are working in a similar way as when the robot is holding the gun. But the tip wear measurement has to be arranged a little different, since it is not possible to use a fix reference plate in this case. To be able to use the same principles for the measurement and the `MeassureWearL` instruction, it has to be done as described in following items:



- Select or create a relatively stable position on the robot held gripper. It shall be possible for the robot to move this point to the fixed gun tip. see an example in the figure above.
- As in the normal case, code sequences with the measuring instructions included have to be prepared in the user program to be used for the reference measurement and for the tip wear measurement.

• We recommend programming the gun calibration instructions in the same code sequences, directly after the tip wear measurement. See *Calibrate - Calibrate the gun* on page 113 and *CalibL/CalibJ - Calibrate the gun during movement'* on page 108.

• The position in the `MeasureWearL` instruction has to be programmed close to the selected position on the gripper, see graphic above.

• When the instruction is executed the fixed gun tip is touched by the gripper. The tip wear of the fixed gun tip is calculated. Currently used TCP value is automatically recalculated and tip wear data in current `gundata` is automatically updated. For more information, see *MeasureWearL - Measure current electrode wear* on page 123.

Method 2: Tip wear calculation and compensation with ReCalcTcp

A RAPID instruction to be used for tip wear calculation and TCP adjustment with this method is available: `ReCalcTCP`.

As the measurement instruction used in method 1, `MeasureWearL` this instruction is used one time for a preparation and then one time after each tip dressing and after the tips have been exchanged. But in this case the instruction is a logical instruction without movements.

When the instruction is executed the tip wear of the fixed gun tip is calculated. The calculations are based on stored information about the total tip wear and the expected relation between the tip wear of the fixed tip and the total tip wear. The tip wear is then compensated in current `tooldata`. Current used TCP value is then automatically recalculated and tip wear data (*curr_wear_fix*) in current `gundata` is automatically updated.

In this case the gun calibration instructions (`Calibrate` or `CalibL/J`) has to be executed before `ReCalcTCP` is used since the total tip wear is used for the calculations.

For more information, see *ReCalcTCP - Recalculate the TCP* on page 131.

The advantages with the calculation method is that it is faster since no extra measurement has to be done, and it is also easier to set up since no reference position has to be arranged for. The disadvantage is that the compensation will not be as accurate as with the measuring method in the cases when the tip wear ratio value not is set to a value in agreement with the reality, or after tip change if the new tips not have the same size every time.

3.7 Setup data for Software Equalizing

There is special setup data for the Software Equalizing functions. The data is found in the SWDEFUSR system module, together with other setup data for the *Spot Servo* functionality. Since SWDEFUSR is a noview module this data is *not* changeable from the FlexPendant. Normally this data is set once during the customizing or installation phase. For more information about general setup data for the *Spot Servo* functionality, see *Customizing* on page 237.

Following setup data is available for the Software Equalizing functions:

Data	Description	Default value
MAX_REL_DIST	Max value for `release_dist` in `gundata` [mm]	15
MAX_DEFLECTION	Max value for `deflection_dist` in `gundata` [mm]	15
m_touch_force	Touch force [N] during tip wear measurement in `MeasureWearL`.	100
teach_dist	Distance from the sheet when teaching weld positions [mm]	0
opposite_z	Defined z direction for the TCP. False is the normal case.	FALSE
MAX_TOUCHUP_STEP	Maximum allowed step value (mm) for weld position touch up.	10
MIN_TOUCHUP_STEP	Minimum allowed step value (mm) for weld position touch up.	0.1
tipwear_sup{SW_NOF_GUNS}	Max allowed digression [mm] in positive and negative direction since last tip wear compensation. Used in `MeasureWearL` and `ReCalcTcp`.	0.2
tipchg_sup{SW_NOF_GUNS}	Max allowed digression [mm] in positive and negative direction from stored reference values. Used in `MeasureWearL`.	3
m_movein_dist	Maximal distance (mm) from programmed point to search for reference surface in `MeasureWearL`.	10

3.8 Limitations

•The functions Gun Arm Release and Deflection Compensation are only activated when *SpotL/J* instructions are used, not for *SpotML/MJ* instructions.

•The data component *pre_close_time* in gundata is not used when software equalizing is active. In this case the preclosing is handled automatically during the movement from the release distance to the weld position.

•For some special configurations an acceptable touch up position can be hard to reach. Then the ReCalcTcp is an alternative method that can be used instead.

•It is not possible to run *SpoL/J* instructions with software equalizing active if independent mode is activated.

•It is not possible to run synchronized Spot instructions when the software equalizing mode is activated. It is only possible to run spot instructions with software equalizing activated in semi coordinated mode.

Note!

Software Equalizing does not work for the SpotML and SpotMJ instructions.

3.9 SoftMove Equalizing

Introduction

This section describes the SoftMove Equalizing method. This function makes it possible to use spot welding guns without mechanical equalizing systems, and it can be used as an alternative to the standard software equalizing functionality. SoftMove Equalizing is activated by selecting an optional switch in the `SpotL` and `SpotJ` instructions, see *SpotL/SpotJ - The basic spot welding instructions* on page 73.

Prerequisites

SoftMove Equalizing is available for the `SpotL` and `SpotJ` instructions, and it is only available if the RobotWare options *Spot Servo Equalizing* (635-3) and *SoftMove* (885-1) are installed.

Function overview

The SoftMove Equalizing works differently than the standard software equalizing method. That is, during the gun closing in a spot instruction the robot will be set to a ‘soft state’ in the tool z direction and remain in that state until the gun starts to open. This method will be more forgiving regarding programming errors than the standard software equalizing method. For example, if the programmed position is not located exactly on the plate but some millimeters away.

Note!

If the normal software equalizing is deactivated then the SoftMove Equalizing will also be deactivated, and it is considered that mechanical equalizing system is being used.

Recommendations

When using SoftMove Equalizing it is important that the tool data is defined correctly, especially the mass. Errors in the load definition will be interpreted as external forces, which in turn can cause the robot to move. Hence, an incorrect definition can cause robot movements.

Some general tips when using SoftMove Equalizing:

- Verify that the load definition of the tool is correct.
- Avoid singular robot orientations.
- If the robot is not soft enough, then try using another arm orientation.

Warning!
When using SoftMove Equalizing it is very important that the load definition of the tool is defined correctly.

For more information about SoftMove, see *Application manual – SoftMove*.

Limitations

The following limitations apply when using SoftMove Equalizing:

- Force offset tuning can only be done in manual operating mode.
- Can not be combined with the option *MultiMove Coordinated* (option 604-1).

When the function is active, that is, during the welding phase in the `SpotL/J` instructions, the following functionality is not accessible:

- *Collision Detection* (option 613-1)

Note!
For more information about SoftMove limitations, see *Application manual – SoftMove*.

Note!
For safety reasons the position supervision limit in x, y, and z directions has been set to 50 mm by default. These values can be changed in the motion parameter configuration if needed, for example type *Motion type CSS* and *Max pos error in z*.

Note!
SoftMove Equalizing is only implemented for the `SpotL` and `SpotJ` instructions and does not work for the `SpotML` and `SpotMJ` instructions.

Programming

An optional parameter must be activated in the spot instructions to use the SoftMove Equalizing functionality (`\SMEQ`). The parameter is an on/off switch for the SoftMove functionality, that is, if the parameter is not used then the normal software equalizing functionality will be used for that instruction, see programming example below.

The optional parameter `\SMEQ` of type `smeqdata` has data components for SoftMove, specific for each position. `Smeqdata` has the following structure:

- Desired *stiffness* (spring effect) of the robot for the specific position.
- Desired *force_offset* (friction compensation) for the specific position.

The *stiffness* parameter defines how strongly the robot tries to move back to the reference point as it is moved away from it, usually works as a low value is good for most positions.

The *force_offset* parameter is needed to compensate for the robot's static friction in a specific position, and the value must be set for each position for the first time the program is executed, see *Force Offset tuning procedure* on page 65.

For more information about this data type, see *4.2.5 smeqdata - SoftMove Equalizing data on page 151*

Example

This program example uses `SpotL` instructions with and without SoftMove Equalizing functionality. Instructions at targets P20 and P40 will be executed with software equalizing while instruction at P30 is executed with SoftMove Equalizing.

```
PERS smeqdata softmdata1:=[1,150]
PROC main()

MoveJ P10, v1000, z50, tool1;
SpotL P20, vmax, gun1, spot11, tool1;
SpotL P30, vmax, gun1, spot12\SMEQ:=softmdata1, tool1;
SpotL P40, vmax, gun1, spot14, tool1;

ENDPROC
```

Execution

When a spot instruction is executed with the `\SMEQ` data set the robot will move to the programmed point and then start to move towards the plate in the tool z direction with an offset of up to 5 mm at the same time as the gun closes.

While the gun is closing the robot will be set into soft state with an additional force offset activated to overcome the friction of the robot. After the weld process is completed the gun will start to open and the soft state will be deactivated, see figure 13 and 14 below.

The SoftMove Equalizing method will be more similar to mechanical equalizing where the gun tips are set into a floating state during the weld.

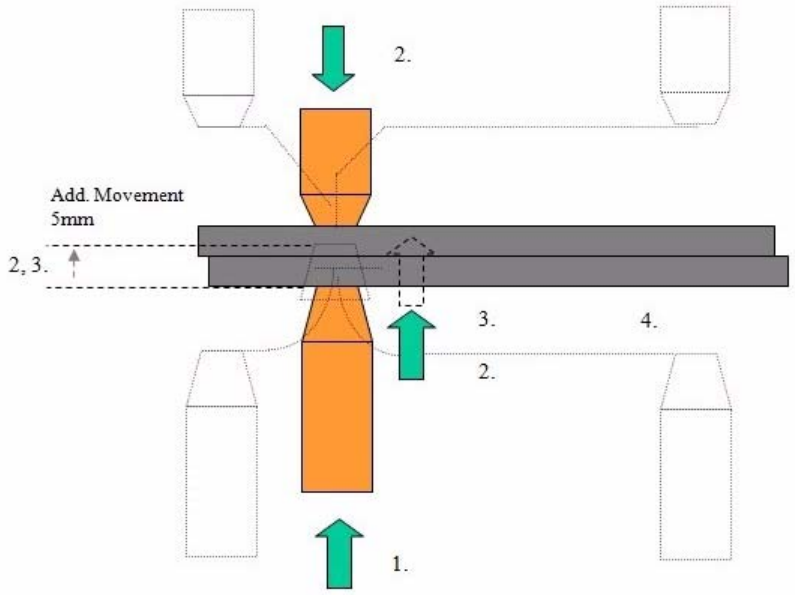


Figure 13 Spot weld sequence on the plate. Position programmed on the sheet.

1. Movement to the programmed point.
2. From the programmed point an additional robot movement (up to 5 mm) is done at the same time as the soft state is activated and the gun closes.
3. When the gun has closed the robot will be “soft” with a small spring effect and the weld will start.
4. After weld the gun will open and the soft state will be deactivated.

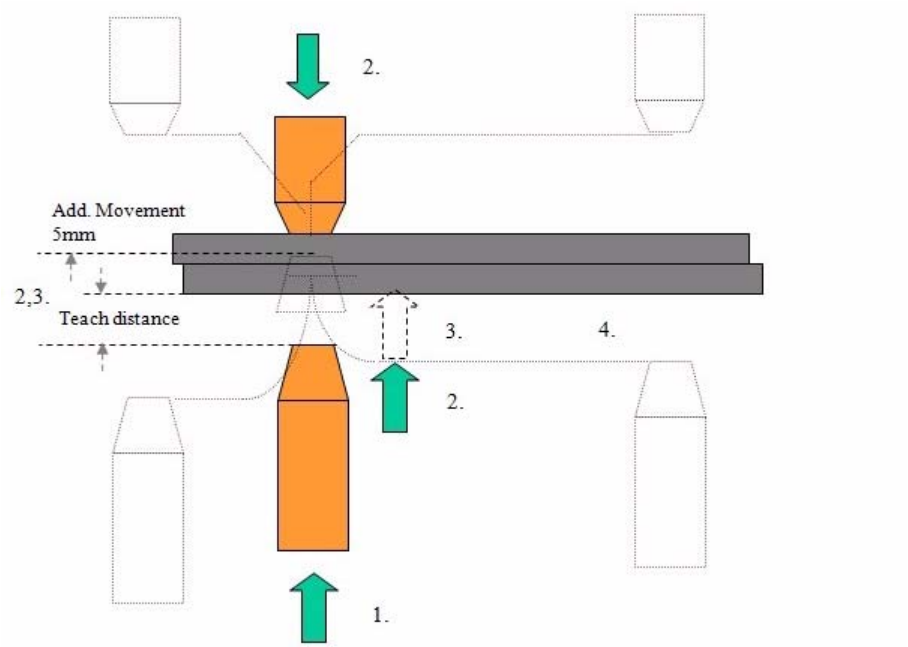


Figure 14 Spot weld sequence outside the plate, with a teach distance set. Position programmed outside the sheet (teach dist).

1. Movement to the programmed point.
2. From the programmed point an additional robot movement (up to 5 mm) + the teach distance, is done at the same tim as the soft state is activated and the gun closes.
3. When the gun has closed the robot will be “soft” with a small spring effect and the weld will start.
4. After weld the gun will open and the soft state will be deactivated.

Force Offset tuning procedure

When a spot instruction is executed the *force_offset* parameter in the current `smeqdata` will be checked, if the value is equal to zero the robot will move to the robtarget specified in the instruction and stop. The operator can tune the *force_offset* value needed to compensate for the friction in the current position. That is, when the parameter value is zero the parameter can be tuned automatically using a dialog on the FlexPendant (internally with CSSOffset-Tune), see figure 15 and 16 below.

If the *force_offset* parameter is greater than zero the spot instruction will be executed normally with the current value to compensate for the friction.

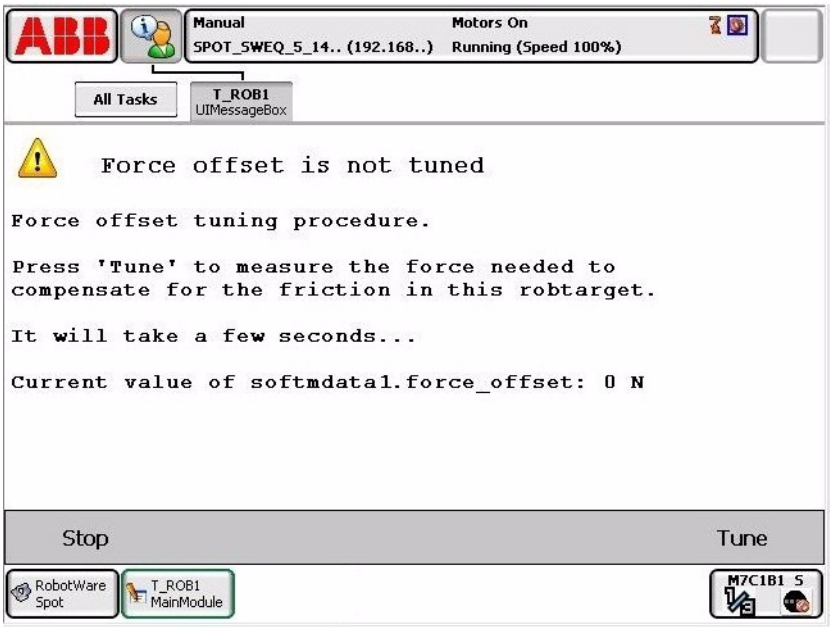


Figure 15 Force offset tuning 1

Stop:

- Stops the program execution.

Tune:

- The force offset will be tuned for the current position. The tuning will take a few seconds and after that a dialog appears with the possibility to accept the measured value or redo the tuning.

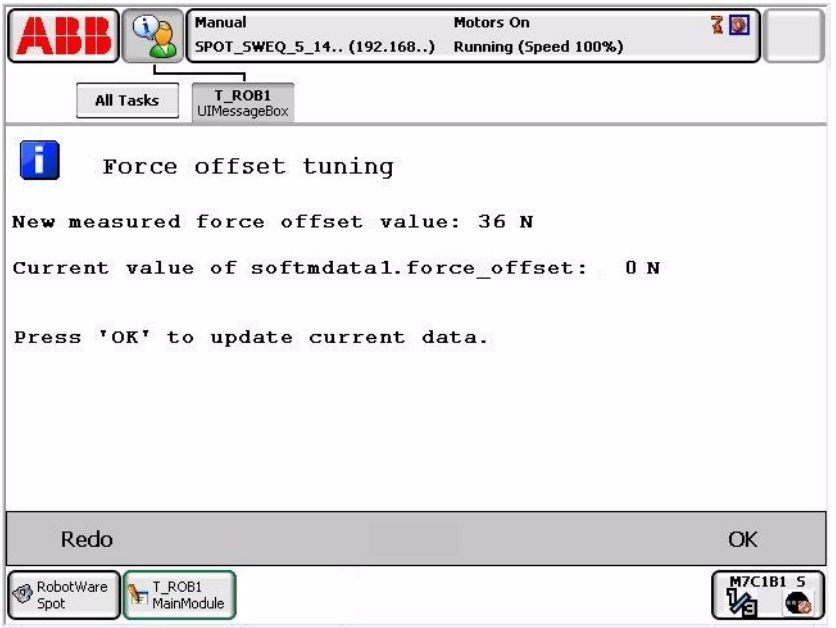


Figure 16 Force offset tuning 2

Redo:

- Returns to the previous screen with the possibility to tune the force offset needed again.

OK:

- Updates the current `smeqdata` with the measured value and the weld will be performed.

Example

Before SpotL softmdata1.force_offset is 0 N.

```
SpotL P30, vmax, gun1, spot12\SMEQ:=softmdata1, tool1;
```

After `SpotL` `softmdata1.force_offset` has been updated to for example 36 N.

Note!

The force offset tuning is only possible to perform in manual mode, not in auto mode.

Note!

The measured value of *force_offset* is only an approximate value and it can be necessary to have a higher or lower value than measured for the current position. For more information about the SoftMove functionality and limitations see *Application manual – SoftMove*.

Tip!

It is also possible to tune the *force_offset* by activating the *Weld position Touch Up* mode (sim_type = 3). If SoftMove Equalizing is active in a `SpotL/J` instruction, an additional button will be visible on the FlexPendant, see figure 17 and 18 below.

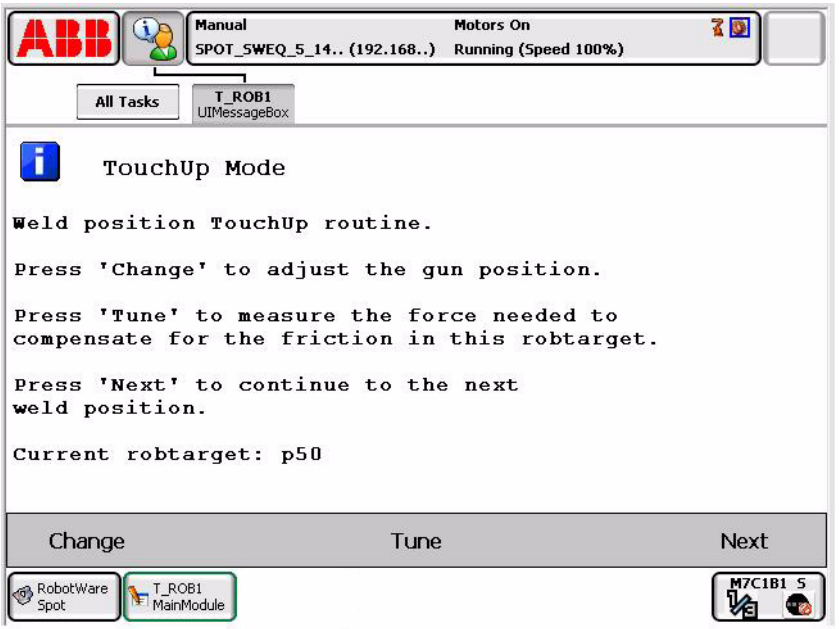


Figure 17 Force offset tuning touch up 1

Change:

- Weld position touch up, see *Weld position Touch Up* on page 45.

Tune:

- Opens the next dialog where the operator can select to perform the force offset tuning or return back to this screen, see below.

Next:

- Moves the robot to the next position, see *Weld position Touch Up* on page 45.

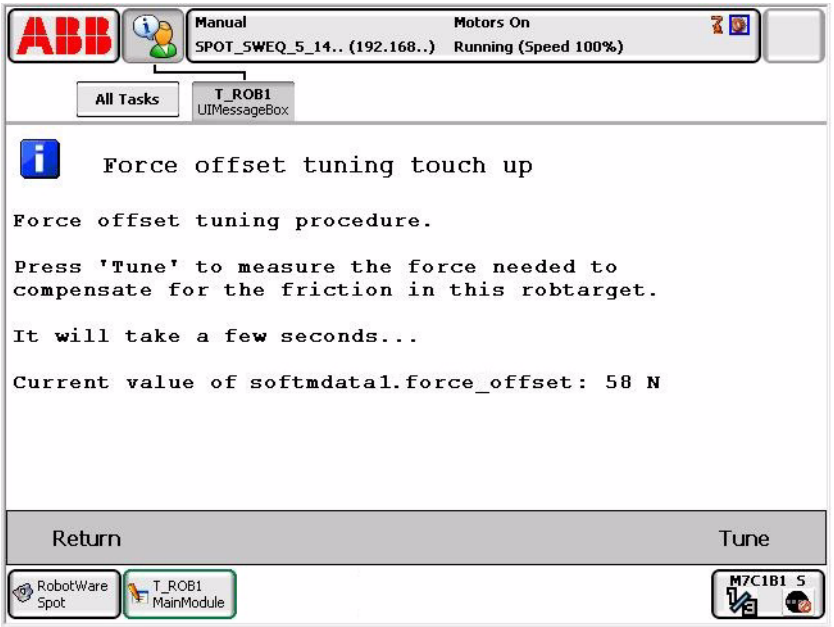


Figure 18 Force offset tuning touch up 2

Return:

- Returns to the previous screen.

Tune:

- The force offset will be tuned for the current position. The tuning will take a few seconds and after that a dialog will appear with the possibility to accept the measured value or return to this screen and redo the tuning.

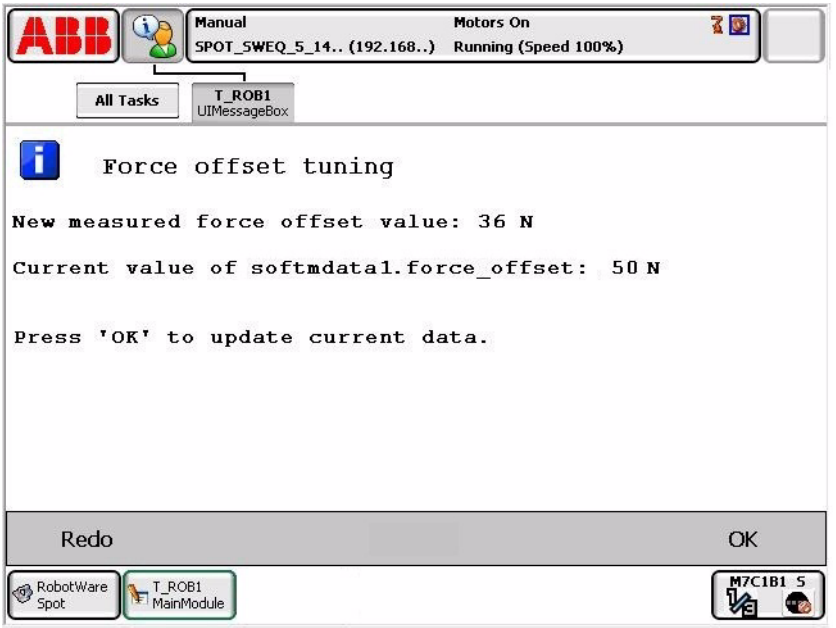


Figure 19 Force offset tuning touch up 3

Redo:

- Returns to the previous screen with the possibility to tune the force offset needed again.

OK:

- Updates the current `smeqdata` with the measured value and goes back to the Touch Up dialog, see figure 17 above.

Example

Before `SpotL softmdata1.force_offset` is 50 N.

```
SpotL P30, vmax, gun1, spot12\SMEQ:=softmdata1, tool1;
```

After `SpotL softmdata1.force_offset` has been updated to for example 36 N.

If auto mode is selected this dialog will be shown, since it is not possible to run the force offset calibration in auto mode.

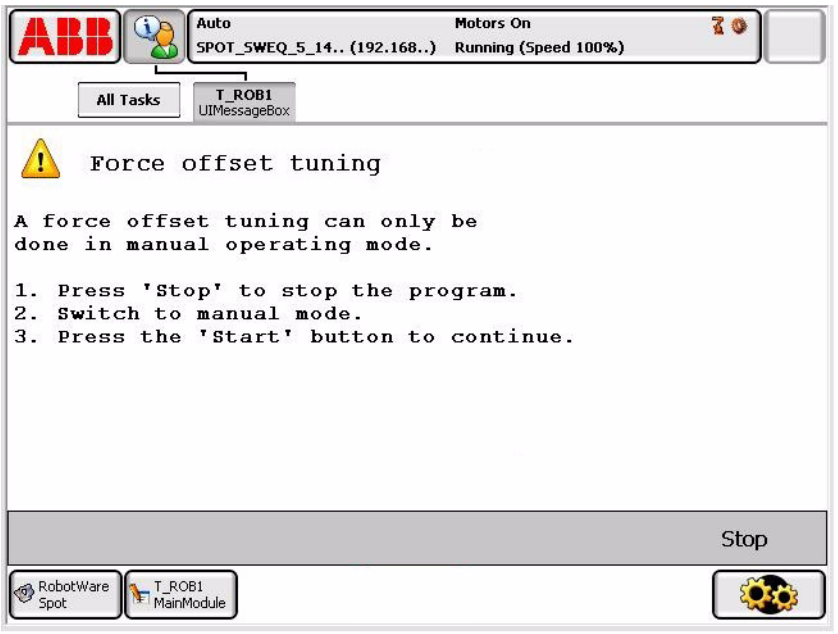


Figure 20 Force offset tuning in auto mode

Stop:

- The program execution will be stopped, and the operator must change operating mode to manual mode and continue the execution from there.

Related information

	Described in
System module - SWDE-FINE	*Service routines (Manual Actions)* on page 164
System module - SWUSER	*Data* on page 154
System module - SWDE-FUSR	*Setup data* on page 168
MeasureWearL	*MeasureWearL - Measure current electrode wear* on page 123
ReCalcTcp	*ReCalcTCP - Recalculate the TCP* on page 131
Customizing	*Customizing* on page 237
SpotL/SpotJ	*SpotL/SpotJ - The basic spot welding instructions* on page 73
SoftMove Equalizing data	*4.2.5 smeqdata - SoftMove Equalizing data on page 151*
SoftMove	Application manual - SoftMove

4 RAPID references

4.1 Instructions

4.1.1 SpotL/SpotJ - The basic spot welding instructions

Descriptions

`SpotL` and `SpotJ` are used in spot welding when welding with one gun or several guns in sequence. The instructions are used to control the complete welding sequences, that is, the motion, gun closure/opening, and the welding process. `SpotL` moves the TCP linearly to the weld position and then activates the weld process. `SpotJ` moves the TCP non-linearly to the weld position before the weld process is activated.

This instruction can only be used in the Main task or, if in a MultiMove system, in Motion tasks.

Example

```
SpotL p100, vmax, gun1, spot10, tool1;
```

This is the only instruction needed to implement a complete welding operation with one gun equipment.

- The TCP for tool1 is moved on a linear path to the position p100 with the speed given in vmax.
- The weld position is always a stop position since the welding is always performed while the robot is standing still.
- The gun is closed in advance when the robot is moved.
- The weld process is started and supervised until finished and the gun is reopened.
- The parameter spot10 is a data of type spotdata containing spot weld specific parameters for the spot in p100, for example desired weld timer program number and gun pressure.
- The parameter gun1 is a num corresponding to the used gun equipment. All gun equipment used are defined in the gundata array curr_gundata in SWUSER.

Arguments

SpotL ToPoint [\ID] Speed GunNo Spot [\InPos] [\OpenHLift] [\CloseHLift] [\QuickRelease] [\SMEQ] Tool [\WObj]

SpotJ ToPoint [\ID] Speed GunNo Spot [\InPos] [\OpenHLift] [\CloseHLift] [\QuickRelease] [\SMEQ] Tool [\WObj]

ToPoint **Data type: robtarget**

The destination point of the robot and additional axes. It is defined as a named position or stored directly in the instruction (marked with an * in the instruction).

[\ID] (Synchronization id) **Data type: identno**

This argument must be used in a MultiMove system, if coordinated synchronized movement, and is not allowed in any other cases.

The specified id number must be the same in all cooperating program tasks. The id number gives a guarantee that the movements are not mixed up at runtime.
For Software Equalizing see the Limitations section.

Speed **Data type: speeddata**

The speed data that applies to movements. Speed data defines the velocity for the tool center point, the tool reorientation and additional axes.

GunNo **Data type: num**

Used gun equipment number. Corresponding to the element number in the gundata array curr_gundata in SWUSER

Spot **Data type: spotdata**

Spot specific data for the weld process.

[\InPos] **Data type: switch**

The optional argument \InPos inhibits the preclosing of the gun. The gun is closed first when the robot has reached the end position. This argument will increase the execution time but is useful in narrow situations.This switch will not affect the execution when software equalizing is active.

[\OpenHLift] **Data type: switch**

The optional argument \OpenHLift will set the gun to it's large gap after the weld. If the argument is omitted the gun opens to it's small gap (work stroke). If the instruction is executed backwards the gun opens to the large position before the motion. (Only valid for pneumatic guns).

[\CloseHLift] **Data type: switch**

The optional argument \CloseHLift will set the gun to it's small gap (work stroke) before closing the gun. If the instruction is executed backwards the gun opens to the large position after the motion. (Only valid for pneumatic guns).

[\QuickRelease] **Data type: switch**

The optional argument \QuickRelease will skip the release movement after the weld if software equalizing is activated.

[\SMEQ] **Data type: softmovedata**

If the optional data \SMEQ is used the robot will be set into a soft state in the tool z direction during the gun closing and the welding phase. This method can be used as an alternative to the 'Gun arm Deflection Compensation' method if the tolerances of the parts to be welded are less exact. For more information see *SoftMove Equalizing* on page 60.

Note!

The SoftMove functionality is only avalible for the option *SoftMove* (885-1).

Tool **Data type: tooldata**

The tool in use when the robot moves. The tool center point is the point moved to the specified destination position, and should be the position for the electrode tips when the gun is closed.

[\WObj] **Data type: wobjdata**

The workobject (coordinate system) to which the robot position in the instruction is related.

This argument can be omitted, and if it is, the position is related to the world coordinate system. If, on the other hand, a stationary TCP or coordinated additional axes are used, this argument must be specified in order to perform a linear movement relative to the work object.

Communication

SpotL/J instructions communicates with the weld equipment using digital signals.
The default I/O setup is located on virtual units (type Virtual).

For a complete description of the I/O configuration, see *I/O configuration* on page 177.

Program execution

Internal process sequence when a SpotL/J instruction is executed:

	Action
1	The robot and gun starts to move towards the programmed position.
2	The weld program number is set. (eg. g1_weld_prog).
3	The gun pressure is set by the weld controller. (Pneumatic guns only).
4	Current spotdata (curr_spotdata in SWUSER) is updated
5	SwInitUserIO is executed.
6	SwPrepare is executed.
7	SwCloseGun is executed, according to the predefined preclosing time in gundata. (Pneumatic guns are closed here).
8	A servo gun will start to close before the position is reached (unless argument \InPos is used), according to the predefined preclosing time in gundata.
9	The equalizing signal is set according to the predefined pre equalizing time in gundata. (eg. g1_equalize).
10	SwPreWeld is executed when the weld position is reached. (Preweld supervision.)
11	The plate thickness is checked. (Servo guns only).
12	The requested gun force is established.
13	The start signal to the weld controller is set. (eg. g1_start_weld)
14	The weld controller performs the weld.
15	When the weld complete signal from the weld controller is received, (eg. g1_weld_complete) a servo gun will start to open.
16	SwOpenGun is executed. (Pneumatic guns are opened here). The equalizing signal is reset.
17	SwPostWeld is executed.
18	The instruction is ready.

Gun closure

The gun closure is activated at a defined time before the weld position, irrespective of the actual speed. The preclose time for each gun equipment are defined in the corresponding gundata in the array curr_gundata in module SWUSER.

For servoguns the movement to the weld position starts with a synchronous phase which means that the servo gun axis is moved synchronized with the robot movement.

The gun closing speed is automatically adapted so the contact position is reached at the same time as the robot reaches the programmed weld position.

If the preclose time is set to 0 in curr_gundata or if the optional argument \InPos is set in current instruction then the gun closure is activated first when the robot has reached the weld position.

Note!

If the optional data \SMEQ is used the robot will be set into a soft state during the gun closing and remain in that state until the weld is completed, for more information see *SoftMove Equalizing* on page 60.

Welding

When the welding position is reached the gun starts to build up the gun force and the user hook SwPreWeld is executed. The plate thickness is checked (Servoguns only). The weld start signal is set as soon as SwPreWeld is ready and the requested gun force is reached. After starting, the system waits for weld complete from the weld equipment.

For pneumatic guns the start signal is set as soon as the robot has reached the weld position and a number of supervisions have been acknowledged. The start signal is high during the entire welding period. It is reset either after weld complete or after a predefined timeout time elapsed.

Gun opening

The gun starts to open to the programmed position after the weld process is finished. At the same time the user hook *SwOpenGun* is executed. When the gun is opened enough and *SwOpenGun* is ready then the movement is released and the robot movement is started.

The gun is also opened to the programmed position after a weld error or in other error situations.

For pneumatic guns the gun opens to a small or large stroke after the welding has finished, depending on the parameter \OpenHLift. The opening is supervised in such a way that the gun open signal is expected.

Program stop and restart

Stop during the motion and restart

The robot stops on the path. If the gun closure already is started the gun will open to the programmed position.

On restart, the robot continues towards the programmed position, closes the gun again and the sequence in *SpotL/J* carries on as normal.

Stop during welding and restart

The welding is finished, validation is done after the stop and the gun opens.

On restart, the robot continues with next instruction.

Quick stop and restart

Quick stop during the motion and restart

The robot stops immediately probably deviated from the path. If the gun closure already is started the gun will open to the programmed position.(Servo guns only).

On restart, the robot first moves back to the path, then continues towards the programmed position, closes the gun again and the sequence in *SpotL/J* carries on as normal.

Quick stop during welding and restart

The weld process is interrupted. The gun is still closed but the gun force will be reduced. (Servo guns only).

A pneumatic gun will open in this situation.

On restart, the weld error handling is executed with possibilities to reweld the last spot.

Instruction by instruction execution

Forward

The instruction is executed in two steps:

	Action
1	The robot is moved to the weld position. After this step it is possible to modify the position.
2	The gun is closed and the weld process is executed.

Backward

The motion is performed backwards to the programmed position with gun control, but the gun is not closed in the weld position and no weld process is activated.(Servoguns only).

For pneumatic guns the gun is set to work or highlift stroke depending on position of the \OpenHLift switch. The motion is performed backwards.

The gun is set to work or highlift stroke depending on the position of the \CloseHLift switch.

Simulated welding

All active simulations are defined in *curr_simdata* in SWUSER.

Weld simulation in the robot controller

Activated by setting *sim_type* = 1.
This will inhibit the start signal to the timer. The simulation time is defined in *sim_time*.
No preweld supervision is performed.

Weld simulation in the timer

Activated by setting *sim_type* = 2.
This will set all enable_current signals low at the next weld.
No preweld supervision is performed.

Testing without closing the guns

Activated by setting *inhib_close* TRUE. Can only be used with sim_type 1 or 2.

Testing without plates

Activated by setting *no_plates* TRUE.
This inhibits the plate thickness supervision. Can only be used with sim_type 1 or 2. (Servo guns only).

Weld position Touch Up mode

Activated by setting sim_type = 3. Only available if Spot Servo Equalizing is installed.
This mode can also be used to update the force_offset if SoftMove Equalizing is activated, for more information see *SoftMove Equalizing* on page 60.

Disable all simulations

All simulations are disabled if sim_type = 0 in curr_simdata.

Error handling

The following error situations can occur:

- Instruction parameter supervision.
- Supervision in the beginning of the movement.
- Gun closure supervision.
- Detection of missing or improper plates (Servo guns only).
- Supervision before weld start.
- Weld error.
- Supervision after welding.
- Gun opening supervision.

Instruction parameter supervision

The error occurs when SpotL/J is called with faulty parameters.

- The signal *process_error for current gun* is set.The program stops.
- An error message is displayed in a dialog box.
- The error message is logged

The parameter must be changed. When the program is restarted the current instruction is restarted from the beginning.

The parameter check_data located in the swdefusr.sys module can be used to decrease the cycle time when running SpotL/J instructions. If set to false the instruction parameter supervision can be turned off.

Supervision in the beginning of the movement

The supervisions in SwPrepare are executing. See *System modules* on page 153.

If an error occurs then:

- The signal process_error for current gun is set.The program stops.
- An error message is displayed in a dialog box with retry possibilities.
- The error message is logged.

Gun closure supervision

Supervisions can be inserted in SwCloseGun. This routine is called when the gun starts to close. See *System modules* on page 153.

For pneumatic guns the supervisions in SwCloseGun are executed. An error occurs if the gun_open signal is not set within a certain time.

- The signal process_error for current gun is set.The program stops.
- An error message is displayed in a dialog box with retry possibilities.
- The error message is logged

For servo guns there are no supervisions in the default version of the routine.

Detection of missing or improper plates (Servoguns only)

An error will be detected by the process kernel if the plate thickness differ more than the allowed limit, defined by the tolerance, from the programmed thickness.

There are three types of errors:

- Negative gun position, one of the tips are missing on the gun, or a tip_wear calibration is needed.
- Missing plates, the plate thickness is smaller than the thickness defined in spotdata.
- Improper geometry, the plate thickness exceeds the tolerance defined in spotdata.

The gun opens.

- The signal process_error for current gun is set.The program stops.
- An error message is displayed in a dialog box with retry possibilities.
- The error message is logged

The following manual choices are available: Ignore / Retry

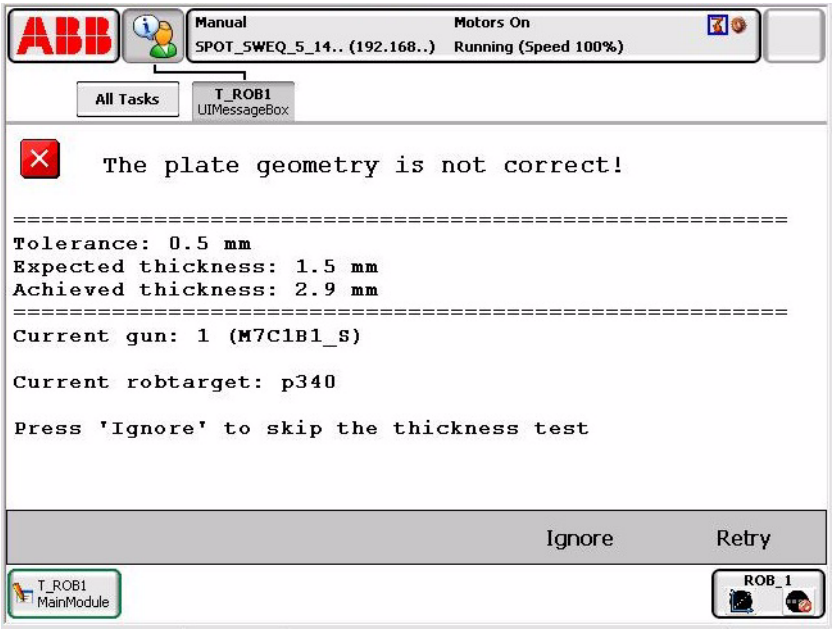


Figure 21 Improper geometry

Ignore:

- Close the gun again but without thickness detection and continue the execution.

Retry:

- Start the interrupted process from the beginning.

If the error is of the type improper geometry there is a possibility to do a retry with a higher force on the gun and complete the current weld, that is. when the plates are not properly fixed together.

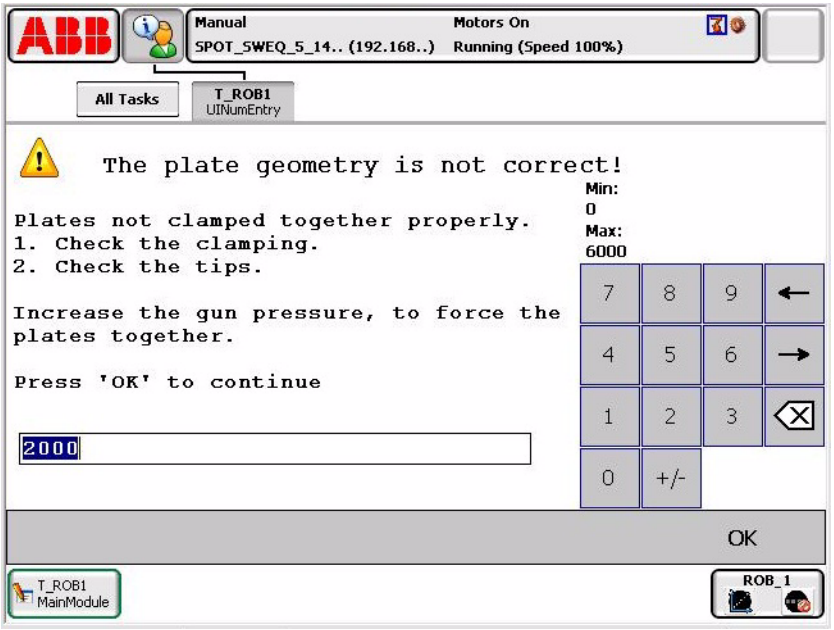


Figure 22 Improper geometry

Supervision before the weld is started

The supervisions in SwPreWeld are executing. See *System modules* on page 153.

If an error occurs then:

- The signal process_error for current gun is set.The program stops.
- An error message is displayed in a dialog box with retry possibilities.
- The error message is logged

Weld error

A weld error occurs either if the weld_fault signal is set during the weld process or if the ready signal from the weld timer has not been set in a certain time (weld_timeout). SpotL/J can be configured to automatically reweld a certain number of times before the error is displayed and the execution stops, waiting for a manual action.

Tip!

The setup parameter *auto_reweld* in swdefusr.sys can be set to the number of welds required.

- The gun opens.
- The signal process_error for current gun is set.The program stops.
- An error message is displayed in a dialog box with retry possibilities.
- The error message and the current robtarget name is logged.
- The following manual choices are available: Info / Skip / Reweld

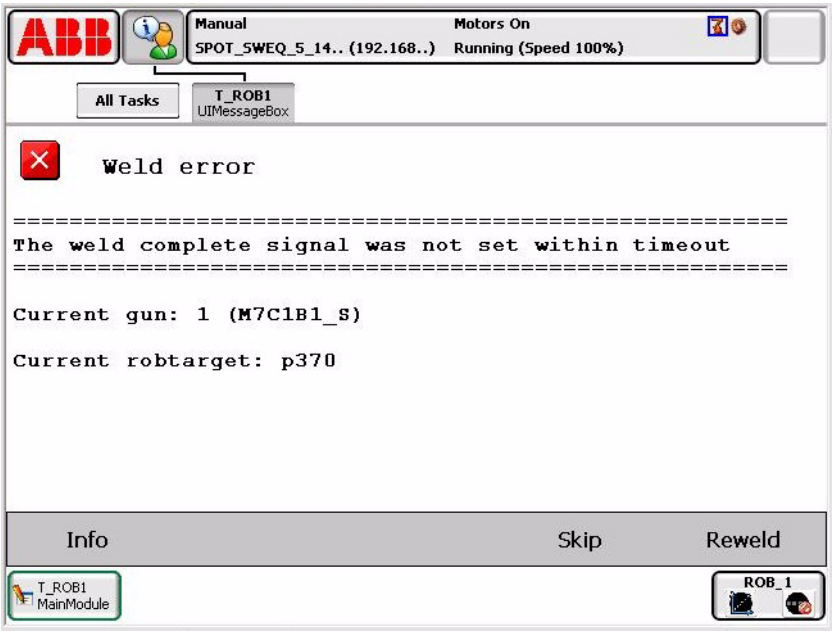


Figure 23 Weld error

Info:

- A dialog box is displayed with more user defined information about the error (if available). By default the current robtarget name will be shown. See user routine SwErrorInfo in *System modules* on page 153.

Skip (Only available in manual mode):

- The reset_fault signal is pulsed.
- The corresponding process error signal is reset.
- The current robtarget name will be stored in the log.The program execution is resumed but omitting the faulty weld.

Reweld:

- The reset_fault signal is pulsed.
- The corresponding process error signal is reset.
- The gun closes.
- The start signal is set after a short time delay and the program execution is resumed.

Skip and Reweld can also be activated by using the digital inputs skip_proc and reweld_proc, see *Basic setup - signal description* on page 179.

Supervision after welding

Supervisions in the user defined routine SwOpenGun are executing. See *System modules* on page 153.

For pneumatic guns the supervisions in SwOpenGun are executed. An error occurs if the gun_open signal is not set within a certain time.

- The signal process_error for current gun is set.The program stops.
- An error message is displayed in a dialog box with retry possibilities.
- The error message is logged

For servo guns there are no supervisions in the default version of the routine.

Gun opening supervision (Servo guns only)

Errors will be detected by internal motion software. An error results in an error message on the TP and a program stop.

User defined error handling

If the user defined error handling is switched on (user_recover = TRUE), the routine *SwErrorRecover* in swuser.sys is called if any of the above error cases occur, except for parameter errors. The input parameters to the *SwErrorRecover* routine carry information about the error case and the chosen error text.

This routine allows customizing the error handling response, that is. the FlexPendant layout and how to resume.

For more information. See *System modules* on page 153.

Example if weld error:

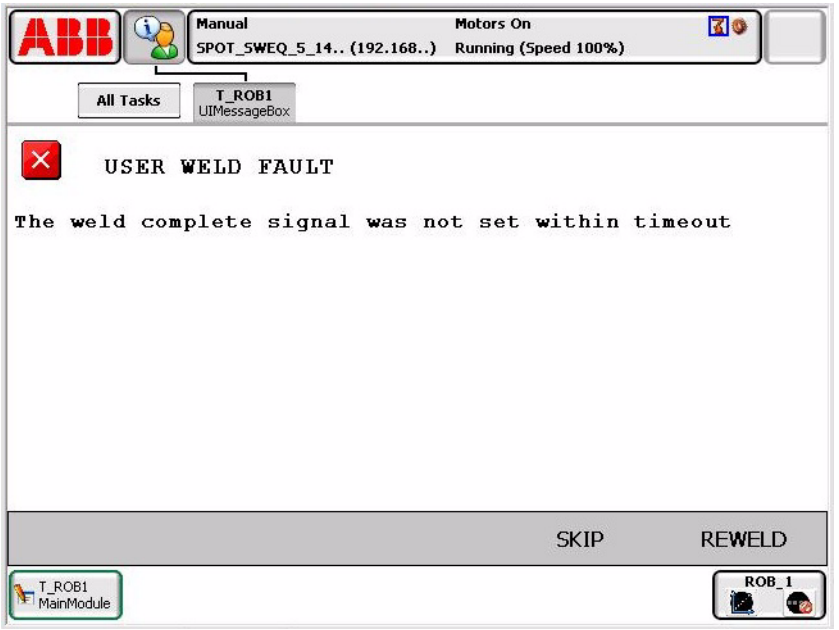


Figure 24 User defined error handling

Power failure handling

At system restart after power failure:

- All spotweld output signals are set to the old status, except the weld start signal.

At program restart after power failure:

- The robot returns to the path and the program execution which was interrupted is continued.
- If a power failure occurred when a weld process was active, the current spot is automatically rewelded.

Software Equalizing

When the software equalizing functions are activated the execution of the SpotL/J instructions is influenced in different ways:

- The movement to the programmed position will be different.
- The gun preclose function is handled automatically.
- The \Inpos switch will not affect the program execution.

For more information, see *Software Equalizing* on page 41.

Customizing

The Spot package gives the user plenty of scope for customizing the Spot functionality. See *Customizing* on page 237, however the main subject of this SpotL/J instruction description is the default setup.

Limitations

Note!

It is not possible use independent gun mode when Software Equalizing is active. This will cause an error message. For more information, see *IndGunMove - Activates independent mode for a servo gun* on page 119.

Note!

It is not possible to run the SpotL/J instruction in coordinated synchronized movement in a MultiMove system if SoftWare Equalizing is active, only semi coordinated movements are allowed.

Note!

The \QuickRelease function is suitable to use if weld positions are located close to each other, not when there is a large distance between weld positions.

Syntax

```
SpotL or SpotJ
[ ToPoint ':=' ] < expression (IN) of robtarget > ','
[ '\' ID ':=' ] < expression (IN) of identno > ','
[ Speed ':=' ] < expression (IN) of speeddata > ','
[ GunNo ':='] < expression (IN) of num > ','
```

```
[ Spot ':='] < persistent (PERS) of spotdata >
[ '\' InPos ]
[ '\' OpenHLift ]
[ '\' CloseHLift ]
[ '\' QuickRelease]
[ '\' SMEQ ':='] < persistent(PERS) of smeqdata > ','
[ Tool ':=' ] < persistent (PERS) of tooldata >
[ '\' WObj ':=' ] < persistent (PERS) of wobjdata > ';'
```

Related information

	Described in:
Definition of velocity	*Technical reference manual - RAPID Instructions, functions and data types - speeddata*
Definition of zonedata	*Technical reference manual - RAPID Instructions, functions and data types - zonedata*
Definition of tool	*Technical reference manual - RAPID Instructions, functions and data types - tooldata*
Definition of work objects	*Technical reference manual - RAPID Instructions, functions and data types - wobjdata*
Definition of spotdata	*spotdata - Spot weld data on page 148*
Definition of gundata	*gundata - Equipment specific weld data on page 138*
SpotML/J	*SpotML/SpotMJ - Spot welding with multiple guns on page 89*
Overview Spot Options	*Summary on page 9*
Customizing possibilities	*Customizing on page 237*
I/O configuration	*I/O configuration on page 177*
Servo gun introduction	*Motion control on page 191*
Servo gun motion parameters	*Application manual - Additional axes and stand alone controller*
Motion in general	*Technical reference manual - RAPID overview*
Software Equalizing	*Software Equalizing on page 41*
SoftMove Equalizing	*3.9 SoftMove Equalizing on page 60*

4.1.2 SpotML/SpotMJ - Spot welding with multiple guns

Description

`SpotML` and `SpotMJ` can be used in spot welding if welding with several guns at the same time is desired. For servo guns it is possible to use two guns simultaneously and for pneumatic guns it is possible to use four guns at the same time. The instructions are used to control the complete welding sequences that is. the motion, gun closure/opening and the welding processes.

- SpotML moves the TCP linearly to the weld position and then activates the gun equipments.
- SpotMJ moves the TCP non-linearly to the weld position before the gun equipment are activated.

These instructions can only be used in the Main task or, if in a MultiMove system, in Motion tasks.

Example

```
SpotML p100, vmax \G1:=spot10 \G2:=spot20, tool1;
```

This is the only instruction needed to implement a complete welding operation with two gun equipment.

- The TCP for tool1 is moved on a linear path to the position p100 with the speed given in vmax. The weld position is always a stop position since the welding is always performed while the robot is standing still. The guns are closed in advance when the robot is moved. The weld processes are started and supervised until finished and the guns are reopened.
- The optional arguments G1 and G2 will activate gun equipment 1 and gun equipment 2. The parameter spot10 is a spotdata containing weld parameters for the welding with gun equipment 1, for example desired weld timer program number and gun pressure. The parameter spot20 contains weld parameters for the welding with gun equipment 2.

All gun equipment used are defined in the gundata array curr_gundata in SWUSER.

Arguments

SpotML ToPoint [\ID] Speed [\G1] [\G2] [\G3] [\G4] [\InPos] [\OpenHLift] [\CloseHLift] Tool [\WObj]

SpotMJ ToPoint [\ID] Speed [\G1] [\G2] [\G3] [\G4] [\InPos] [\OpenHLift] [\CloseHLift] Tool [\WObj]

ToPoint **Data type: robtarget**

The destination point of the robot and additional axes. It is defined as a named position or stored directly in the instruction (marked with an * in the instruction). This name will be stored in the log if a error occurs during the welding.

[\ID] (Synchronization id) **Data type: identno**

This argument must be used in a MultiMove system, if coordinated synchronized movement, and is not allowed in any other cases.

The specified id number must be the same in all cooperating program tasks. The id number gives a guarantee that the movements are not mixed up at runtime.

Speed **Data type: speeddata**

The speed data that applies to movements. Speed data defines the velocity for the tool center point, the tool reorientation and additional axes.

[G1] - [G4] **Data type: spotdata**

Spot data with the spot specific data associated with the weld with gun equipment 1 - 4 for pneumatic guns and with gun equipment 1 - 2 for servo guns.

[\InPos] **Data type: switch**

The optional argument \InPos inhibits the preclosing of the guns. The guns are closed first when the robot has reached the end position. This argument will increase the execution time but is useful in narrow situations.

[\OpenHLift] **Data type: switch**

The optional argument \OpenHLift will set the guns to its large gap after the weld. If the argument is omitted the guns opens to its small gap (work stroke). If the instruction is exe-

cuted backwards the guns opens to the large position before the motion. (Only valid for pneumatic guns).

[\CloseHLift] **Data type: switch**

The optional argument \CloseHLift will set the guns to its small gap (work stroke) before closing the guns. If the instruction is executed backwards the guns opens to the large position after the motion. (Only valid for pneumatic guns).

Tool **Data type: tooldata**

The tool in use when the robot moves. The tool center point is the point moved to the specified destination position, and should be the position for the electrode tips when the gun is closed.

[\WObj] **Data type: wobjdata**

The work object (coordinate system) to which the robot position in the instruction is related.

This argument can be omitted, and if it is, the position is related to the world coordinate system. If, on the other hand, a stationary TCP or coordinated additional axes are used, this argument must be specified in order to perform a linear movement relative to the work object.

Communication

SpotML/MJ instructions communicates with the weld equipment using digital signals. The default I/O setup is located on virtual units (type Virtual).

For a complete description of the I/O configuration, see *I/O configuration* on page 177.

Program execution

Internal process sequence when a SpotL/J instruction is executed:

	Action
1	The robot and gun starts to move towards the programmed position.
2	The weld program number is set. (eg. g1_weld_prog).
3	The gun pressure is set by the weld controller. (Pneumatic guns only).
4	SwInitUserIO is executed.
5	SwPrepare is executed.
6	SwCloseGun is executed, according to the predefined preclosing time in gundata. (Pneumatic guns are closed here).
7	A servo gun will start to close before the position is reached (unless argument \InPos is used), according to the predefined preclosing time in gundata.
8	The equalizing signal is set according to the predefined pre equalizing time in gundata. (eg. g1_equalize).
9	Current spotdata (curr_spotdata in SWUSER) is updated.
10	SwPreWeld is executed when the weld position is reached. (Preweld supervision.)
11	The plate thickness is checked. (Servo guns only).
12	The requested gun force is established.
13	The start signal to the weld controller is set. (eg. g1_start_weld)
14	The weld controller performs the weld.
15	When the weld complete signal from the weld controller is received, (eg. g1_weld_complete) a servo gun will start to open.
16	SwOpenGun is executed. (Pneumatic guns are opened here). The equalizing signal is reset.
17	SwPostWeld is executed.
18	The instruction is ready.

Gun closure

The gun closure is activated at a defined time before the weld position, irrespective of the actual speed. The preclose time for each gun equipment are defined in the corresponding gundata in the array curr_gundata in module SWUSER.

For servoguns the movement to the weld position starts with a synchronous phase which means that the servo gun axis is moved synchronized with the robot movement.

The gun closing speed is automatically adapted so the contact position is reached at the same time as the robot reaches the programmed weld position.
If the preclose time is set to 0 in curr_gundata or if the optional argument \InPos is set in current instruction then the gun closure is activated first when the robot has reached the weld position.

Welding

When the welding position is reached each gun starts to build up the gun force and the user hook SwPreWeld is executed. The plate thickness is checked (Servoguns only). The weld start signal for each process is set as soon as SwPreWeld is ready and the requested gun force is reached. After starting, the system waits for weld complete from the weld equipment.
For pneumatic guns the start signal is set as soon as the robot has reached the weld position and a number of supervisions have been acknowledged.
The start signal is high during the entire welding period. It is reset either after weld complete or after a predefined timeout time elapsed.

Gun opening

Each activated gun starts to open to the programmed position after the welding has finished. At the same time the user hook SwOpenGun is executed. When the guns are opened enough and SwOpenGun is ready then the movement is released and the robot movement is started. The guns are also opened to the programmed position after a weld error or in other error situations.
For pneumatic guns the guns opens to a small or large stroke after the welding has finished, depending on the parameter \OpenHLift. The opening is supervised in such a way that the gun open signal is expected.

Program stop and restart

Stop during the motion and restart

The robot stops on the path. If the gun closure already is started the guns will open to the programmed position.
On restart, the robot continues towards the programmed position, closes the guns again and the sequence in SpotML/MJ carries on as normal.

Stop during welding and restart

The welding is finished, validation is done after the stop and the guns opens.
On restart, the robot continues with the next instruction.

Quick stop and restart

Quick stop during the motion and restart

The robot stops immediately probably deviated from the path. If the asynchronous gun closure already is started the gun will open to the programmed position (Servo guns only).
On restart, the robot first moves back to the path, then continues towards the programmed position, closes the guns again and the sequence in SpotML/MJ carries on as normal.

Quick stop during welding and restart

The weld process is interrupted. The gun is still closed but the gun force will be reduced.(Servo guns only).
A pneumatic gun will open in this situation.
On restart, the weld error handling is executed with possibilities to reweld the last spot.

Instruction by instruction execution

Forwards

The instruction is executed in two steps:

- The robot is moved to the weld position. After this step it is possible to modify the position.
- The guns are closed and the weld processes are executed.

Backwards

The motion is performed backwards to the programmed position with gun control, but the gun is not closed in the weld position and no weld process is activated (Servoguns only).
For pneumatic guns the gun is set to work or highlift stroke depending on position of the \OpenHLift switch. The motion is performed backwards.
The gun is set to work or highlift stroke depending on the position of the \CloseHLift switch.

Simulated welding

All active simulations are defined in curr_simdata in SWUSER. The simulations influences all active weld processes.

Weld simulation in the robot controller

Activated by setting sim_type = 1. This will inhibit the start signal to the timer. The simulation time is defined in sim_time. No preweld supervision is performed.

Weld simulation in the timer

Activated by setting sim_type = 2. This will set all enable_current signals low at the next weld. No preweld supervision is performed.

Testing without closing the guns

Activated by setting inhib_close TRUE. Can only be used with sim_type 1 or 2.

Testing without plates

Activated by setting no_plates TRUE. This inhibits the plate thickness supervision. Can only be used with sim_type 1 or 2 (Servo guns only).

Disable all simulations

All simulations are disabled if sim_type = 0 in curr_simdata.

Error handling

The following error situations can occur:

- Instruction parameter supervision.
- Supervision in the beginning of the movement.
- Gun closure supervision.
- Detection of missing or improper plates (Servo guns only).
- Supervision before weld start.
- Weld error.
- Supervision after welding.
- Gun opening supervision.

Instruction parameter supervision

The error occurs when SpotML/J is called with faulty parameters.

- The signal process_error for current gun is set.
- The program stops.
- An error message is displayed in a dialog box.
- The error message is logged
- The parameter must be changed.
- When the program is restarted the current instruction is restarted from the beginning.
- The parameter check_data located in the swdefusr.sys module can be used to decrease the cycle time when running SpotL/J instructions. If set to false the instruction parameter supervision can be turned off.

Supervision in the beginning of the movement

The supervisions in SwPrepare are executing. See *System modules* on page 153.

If an error occurs then:

- The signal process_error for current gun is set.The program stops.
- An error message is displayed in a dialog box with retry possibilities.
- The error message is logged

Gun closure supervision

Supervisions can be inserted in SwCloseGun. This routine is called when the gun starts to close. See *System modules* on page 153. For pneumatic guns the supervisions in SwCloseGun are executed. An error occurs if the gun_open signal is not set within a certain time.

- The signal process_error for current gun is set.The program stops.
- An error message is displayed in a dialog box with retry possibilities.
- The error message is logged

Detection of missing or improper plates (Servoguns only)

An error will be detected by the process kernel if the plate thickness differ more than the allowed limit, defined by the tolerance, from the programmed thickness.

There are three types of errors:

- Negative gun position, one of the tips are missing on the gun, or a tip_wear calibration is needed.
- Missing plates, the plate thickness is smaller than the thickness defined in spotdata.
- Improper geometry, the plate thickness exceeds the tolerance defined in spotdata.

The gun opens.

- The signal process_error for current gun is set.The program stops.
- An error message is displayed in a dialog box with retry possibilities.
- The error message is logged.

The following maual choices are available: Ignore / Retry

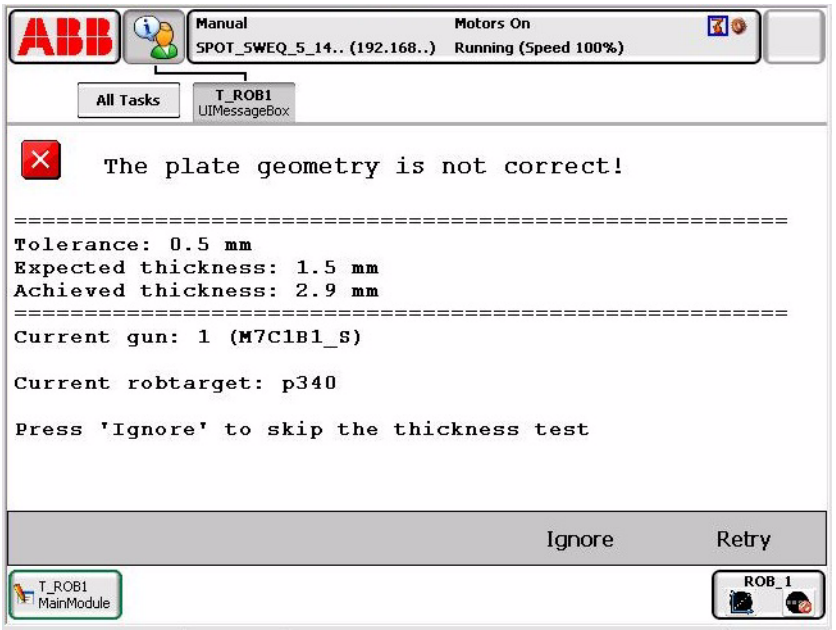


Figure 25 Improper geometry.

Ignore:

- Close the gun again but without thickness detection and continue the execution.

Retry:

Start the interrupted process from the beginning.

If the error is of the type improper geometry there is a possibility to do a retry with a higher force on the gun and complete the current weld, that is. when the plates are not properly fixed together.

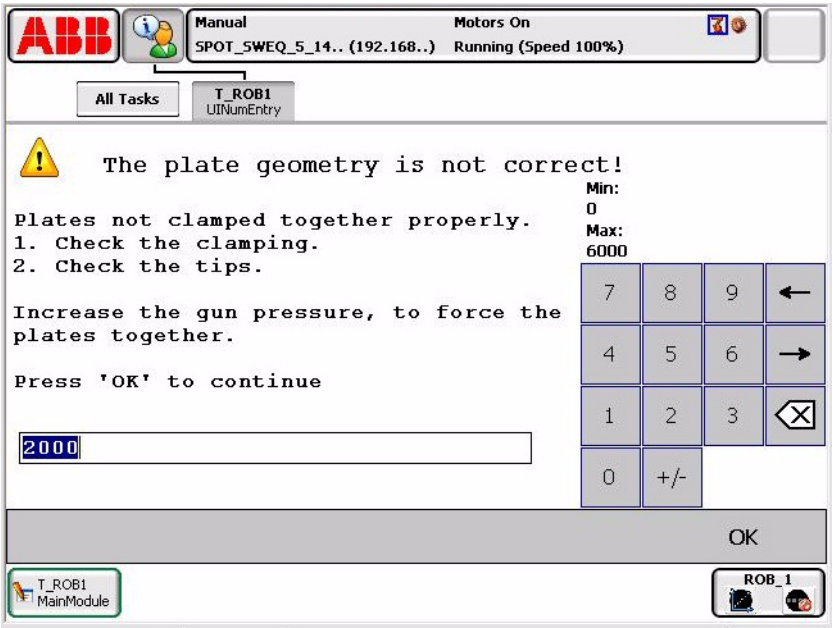


Figure 26 Improper geometry

Supervision before the weld is started

The supervisions in SwPreWeld are executing. See *System modules* on page 153.

If an error occurs then:

- The signal process_error for current gun is set.The program stops.
- An error message is displayed in a dialog box with retry possibilities.
- The error message is logged.

Weld error

A weld error occurs either if the weld_fault signal is set during the weld process or if the ready signal from the weld timer has not been set in a certain time (weld_timeout). SpotML/J can be configured to automatically reweld a certain number of times before the error is displayed and the execution stops, waiting for a manual action.

Tip!

The setup parameter *auto_reweld* in swdefusr.sys can be set to the number of welds required.

- The gun opens.
- The signal process_error for current gun is set.The program stops
- An error message is displayed in a dialog box with retry possibilities
- The error message is logged

The following manual choices are available: Info / Skip / Reweld

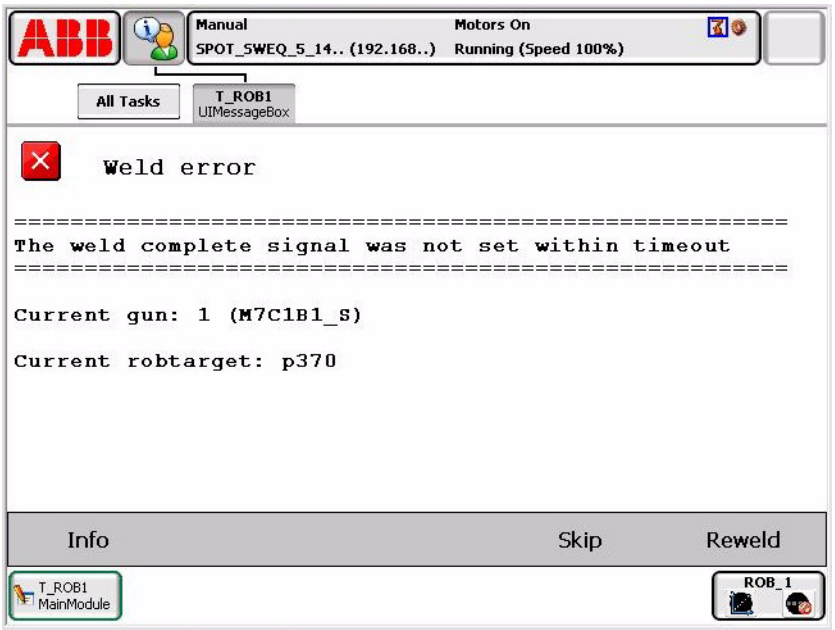


Figure 27 Dialog box for weld error

Info:

- A dialog box is displayed with more user defined information about the error (if available). See user routine SwErrorInfo in *System modules* on page 153.

Skip (Only available in manual mode):

- The reset_fault signal is pulsed.
- The corresponding process error signal is reset.
- The program execution is resumed but omitting the faulty weld.

Reweld:

- The reset_fault signal is pulsed.
- The corresponding process error signal is reset.
- The gun closes.
- The start signal is set after a short time delay and the program execution is resumed.

Skip and Reweld can also be activated by using the digital inputs *skip_proc* and *reweld_proc*, see *I/O configuration* on page 177.

Supervision after welding

Supervisions in the user defined routine SwOpenGun are executing. See *System modules* on page 153. For pneumatic guns the supervisions in SwOpenGun are executed. An error occurs if the gun_open signal is not set within a certain time.

- The signal process_error for current gun is set.The program stops.
- An error message is displayed in a dialog box with retry possibilities.
- The error message is logged.

For servo guns there are no supervisions in the default version of the routine.

Gun opening supervision (Servo guns only)

Errors will be detected by internal motion software. An error results in an error message on the FlexPendant and a program stop.

User defined error handling

If the user defined error handling is switched on (user_recover = TRUE), the routine *SwErrorRecover* in swuser.sys is called if any of the above error cases occur, except for parameter errors. The input parameters to the *SwErrorRecover* routine carry information about the error case and the chosen error text.

This routine allows customizing the error handling response, that is. the FlexPendant layout and how to resume. For more information. See *System modules* on page 153.

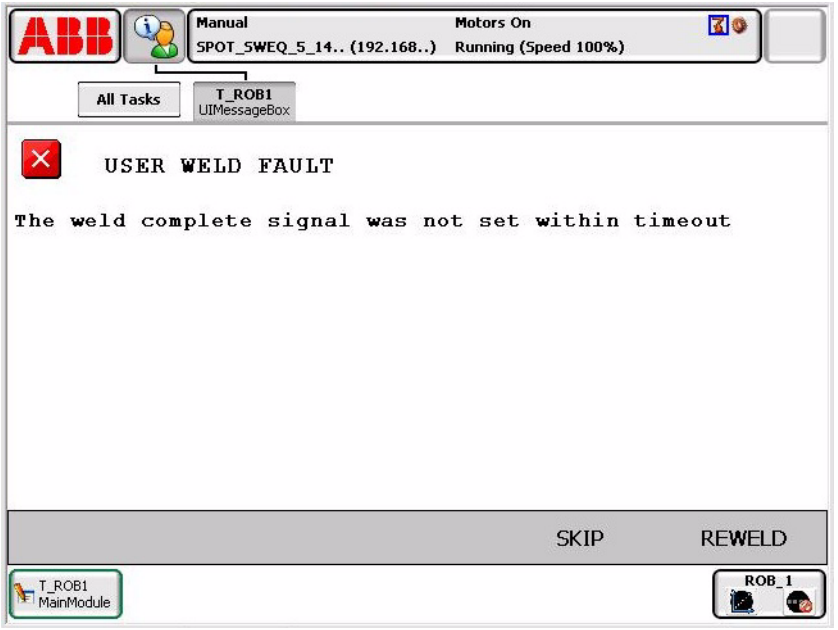


Figure 28 User defined error handling

Power failure handling

At system restart after power failure:

- All spotweld output signals are set to the old status, except the weld start signal.

At program restart after power failure:

- The robot returns to the path and the program execution which was interrupted is continued.
- If a power failure occurred when a weld process was active, the current spot is automatically rewelded.

Customizing

The Spot package gives the user plenty of scope for customizing the Spot functionality. See *Customizing* on page 237.

However, the main subject of this SpotML/J instruction description is the default setup.

Limitations

Note!

It is not possible use Software Equalizing mode for this instruction, SpotML/SpotMJ.

For more information, see *Software Equalizing* on page 41.

Syntax

```
SpotML or SpotMJ
[ ToPoint ':=' ] < expression (IN) of robtarget > ','
[ '\' ID ':=' < expression (IN) of identno >]','
[ Speed ':=' ] < expression (IN) of speeddata > ','
[ '\'G1 ':=' < persistent (PERS) of spotdata > ]
[ '\' G2 ':=' < persistent (PERS) of spotdata > ]
[ '\' G3 ':=' < persistent (PERS) of spotdata > ]
[ '\' G4 ':=' < persistent (PERS) of spotdata > ]
[ '\' InPos ]
[ '\' OpenHLift ]
[ '\' CloseHLift ]','
[ Tool ':=' ] < persistent (PERS) of tooldata >
[ '\' WObj ':=' < persistent (PERS) of wobjdata > ] ';'
```

Related information

	Described in:
Definition of velocity	*Technical reference manual - RAPID Instructions, functions and data types - speeddata*
Definition of zonedata	*Technical reference manual - RAPID Instructions, functions and data types - zonedata*
Definition of tool	*Technical reference manual - RAPID Instructions, functions and data types - tooldata*
Definition of work objects	*Technical reference manual - RAPID Instructions, functions and data types - wobjdata*
Definition of spotdata	*See section spotdata - Spot weld data on page 148*
Definition of gundata	*gundata - Equipment specific weld data on page 138*
SpotL/J	*SpotL/SpotJ - The basic spot welding instructions on page 73*
Overview Spot Options	*Summary on page 9*
Customizing possibilities	*Customizing on page 237*
I/O configuration	*I/O configuration on page 177*
Servo gun introduction	*Motion control on page 191*
Servo gun motion parameters	*Application manual - Additional axes and stand alone controller*
Motion in general	*Technical reference manual - RAPID overview*

4.1.3 SetForce - Close the gun with desired force

Description

`SetForce` is used in spot welding to close the gun and apply a predefined force during a desired time without activating a weld process. The gun will open again after the elapsed time or when a digital input signal is set. The instruction can for example be used for tip dressing.

Example

```
SetForce  gun1, force10;
```

Forcedata force10 contains the parameters for the SetForce action, for example desired tip force and force time.

The parameter gun1 is a num corresponding to the used gun equipment. All gun equipment used are defined in the gundata array curr_gundata in SWUSER.

Arguments

SetForce GunNo [\RetThickness] Force

GunNo **Data type: num**

Used gun number. Corresponding to the element number in the gundata array curr_gundata in SWUSER

[\RetThickness] **Data type: num**

Optional parameter. The achieved thickness [mm]. (Only valid for servoguns).

Force **Data type: forcedata**

The forcedata with the force parameters.

Program execution

Internal sequence when a SetForce instruction is executed:

1. The gun is closed.
2. The plate thickness is checked.(Only valid for servoguns).
3. The requested gun force is established.
4. Wait until the desired force time elapsed or the force complete signal is activated.
5. The gun is opened to the previous position.
6. The force data in the array curr_forcedata in SWUSER is updated.

The force complete signal for each used gun is predefined in the I/O configuration.

For a complete description of the I/O configuration, see *I/O configuration* on page 177.

Error handling

Instruction parameter supervision

The error occurs when SetForce is called with faulty parameters. The program stops.

The parameter must be changed. When the program is restarted the current instruction is restarted from the beginning.

Detection of missing or improper plates (Only for servoguns)

An error will be detected by the process kernel if the plate thickness differ more than the allowed limit defined by the tolerance from the programmed thickness.

There are three different types of errors:

- Negative gun position, one of the tips are missing on the gun, or a tip_wear calibration is needed.
- Missing plates, the plate thickness is smaller than the thickness defined in forcedata.
- Improper geometry, the plate thickness exceeds the tolerance defined in forcedata.

1. The gun opens.
2. The signal process_error for current gun is set.The program stops.
3. An error message is displayed in a dialog box with retry possibilities.
4. The error message is logged.
5. The following manual choices are available: Ignore / Retry / Skip.

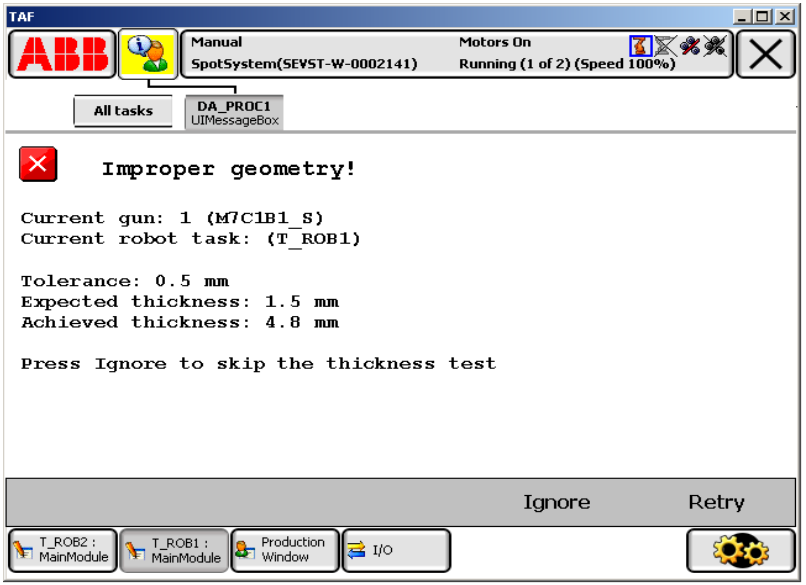


Figure 29 Dialog box for tip positions error.

Ignore:

Close the gun again but without thickness detection and continue the execution.

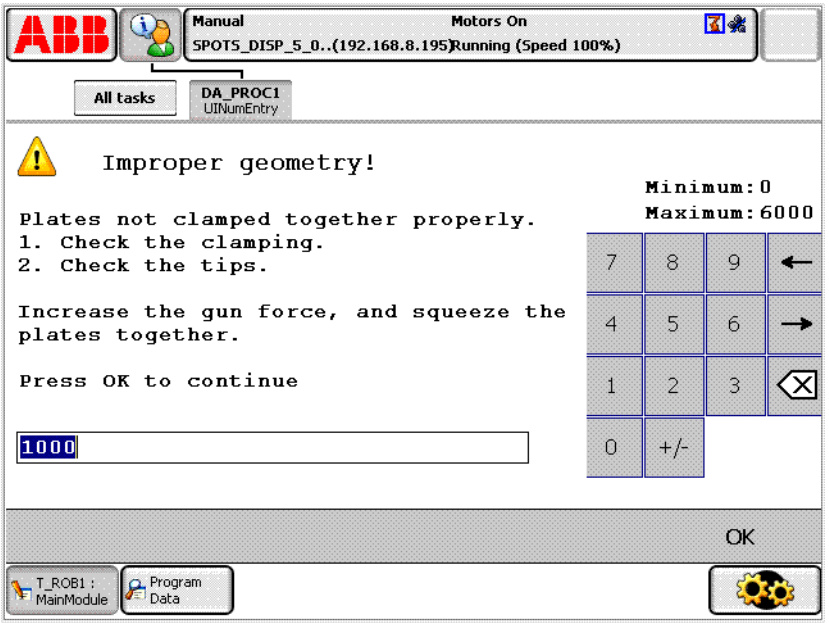
Retry:

- Retry the SetForce instruction.

Skip: (Only available in manual mode)

- Cancel current instruction and continue with next.

If the error is of the type improper geometry there is a possibility to do a retry with a higher force on the gun and complete the current weld, that is. when the plates are not properly fixed together.



Syntax

```
Servo guns:
SetForce
[ GunNo ':='] < expression (IN) of num >
['\' RetThickness ':=' < variable or persistent(INOUT) of num > ]','
[ Force ':='] < persistent (PERS) of forcedata > ';'

Pneumatic guns:
SetForce
[ GunNo ':='] < expression (IN) of num >
[ Force ':='] < persistent (PERS) of forcedata > ';'
```

4.1.4 CalibL/CalibJ - Calibrate the gun during movement'

Description

`CalibL/J` is used in spot welding to calibrate the distance between the gun tips for servo guns. This is necessary after tip change or tool change and it is recommended after welding of a number of spots or performing a tip dress. Calibrate will also update the tip wear data in the used gundata. The calibration is done during a robot movement to a programmed position. NB: The gun performs two non-synchronized close/open movements during the calibration. This instruction can only be used in the Main task or, if in a MultiMove system, in Motion tasks. It can not be executed in synchronized mode, only independent mode is available.

If the option *Spot Servo Equalizing* is installed there are additional error handling included for lost tips when a TipChg calibration is done and supervision of the tip wear when a Tip-Wear calibration is done.

Example

```
CalibL  p400, v500, gun1\ TipWear, fine, tool1;
```

- The gun gun1 is calibrated for TipWear during the linear movement to p400.
- The parameter gun1 is a num corresponding to the used gun equipment. All gun equipment used are defined in the gundata array curr_gundata in SWUSER.
- The data curr_tip_wear in curr_gundata will be automatically updated.

Arguments

CalibL ToPoint Speed GunNo [\TipChg] | [\ToolChg] | [\TipWear] [\RetTipWear] [\RetPosAdj] [\PrePos] Zone Tool [\WObj]

CalibJ ToPoint Speed GunNo [\TipChg] | [\ToolChg]| [\TipWear] [\RetTipWear] [\RetPosAdj] [\PrePos] Zone Tool [\WObj]

ToPoint **Data type: robtarget**

The destination point of the robot and additional axes. It is defined as a named position or stored directly in the instruction (marked with an * in the instruction).

A movement of the gun tip position can not be programmed. This will cause an error message.

Speed **Data type: speeddata**

The speed data that applies to movements. Speed data defines the velocity for the tool center point, the tool reorientation and additional axes.

GunNo **Data type: num**

Used gun equipment number. Corresponding to the element number in the *gundata* array *curr_gundata* in SWUSER

[\TipChg] **Data type: switch**

Calibration type. This calibration type is used after tip change.
The gun will close and open two times. The first close movement will be slow to find the unknown contact position. The total tip wear is reset to zero.
If the option *Spot Servo Equalizing* is selected the difference since last calibration will be supervised. If the difference since the last calibration exceeds the supervision value in the *tipchg_superv* an error will be raised.
See *Setup data for Software Equalizing* on page 58.

[\ToolChg] **Data type: switch**

Calibration type. This calibration type is used after tool change.
The gun will close and open two times. The first close movement will be slow to find the unknown contact position. The total tip wear will remain unchanged.

[\TipWear] **Data type: switch**

Calibration type. This calibration type is used to update the tip wear and adjust the contact position after tip dress or after welding a number of spots.
The gun will close and open fast two times. The total tip wear is updated.
If the option *Spot Servo Equalizing* is selected the difference since last calibration will be supervised. If the difference since the last calibration exceeds the supervision value in the *tipwear_superv* an error will be raised.
See *Setup data for Software Equalizing* on page 58.

[\RetTipWear] **Data type: num**

The achieved tip wear [mm].

[\RetPosAdj] **Data type: num**

The positional adjustment since the last calibration [mm].

[\PrePos] **Data type: num**

The position to move with high speed to before search for contact position with slower speed is started [mm].

Zone **Data type: zonedata**

Zone data for the movement. Zone data describes the size of the generated corner path.

Tool **Data type: tooldata**

The tool in use when the robot moves. The tool center point is the point moved to the specified destination position, and should be the position for the electrode tips when the gun is closed.

[\WObj] **Data type: wobjdata**

The work object (coordinate system) to which the robot position in the instruction is related. This argument can be omitted, and if it is, the position is related to the world coordinate system. If, on the other hand, a stationary TCP or coordinated additional axes are used, this argument must be specified in order to perform a linear movement relative to the work object.

Program execution

Internal sequence when a CalibL/J instruction is executed:

- The robot starts the movement to the destination position.
- The gun will close and open two times during the robot movement. Different tip speeds depending on selected calibration type.
- The gun is opened to the previous position.
- For certain calibration types: curr_tip_wear in the array curr_gundata in SWUSER is updated.

Instruction by instruction execution

Forward

As during continuous execution.

Backward

The motion is performed backwards to the programmed position, but no calibration is activated. NB, the tip distance in this case is the programmed value in the instruction

Positional adjustment

The optional argument RetPosAdj can be used to detect if for example the tips are lost after a tip change. The parameter will hold the value of the positional adjustment since the last calibration. The value can be negative or positive.

If the option *Spot Servo Equalizing* is selected this value will be used to calculate the difference since last calibration and supervise the tips when calibrating.

Using a pre position

In order to speed up the calibration, it is possible to define a pre position.When the calibration starts, the gun arm will be run fast to the pre position, stop and then continue slowly forward in order to detect the tip contact position. A pre position will be ignored if it is larger than the current gun position (in order not to slow down the calibration).

Error handling

Instruction parameter supervision

The error occurs when *CalibL/J* is called with faulty parameters or if no calibration type switch is programmed. The program stops with error text.

The parameter must be changed. When the program is restarted the current instruction is restarted from the beginning.

Tip change supervision:

If the calculated difference to the last calibration of the gun exceeds the supervision value defined in the setup data *tipchg_superv* an error will be raised and the program execution will be stopped. This error can occur for example after tip change and when CalibL/J ... \TipChg is called with wrong (too large or too small tips) tips. The program stops with error message.

This error handling only exists for the *Spot Servo Equalizing option.*

Tip wear supervision:

If the calculated difference to the last calibration of the gun exceeds the supervision value defined in the setup data *tipwear_superv* an error will be raised and the program execution will be stopped. This error can occur for example after tip dressing when CalibL/J .. \TipWear is called with badly dressed tips. The program stops with error message.

This error handling only exists for the *Spot Servo Equalizing option.*

Limitations

It is not possible to execute CalibL/J instructions in coordinated synchronized movement in a MultiMove system.

Syntax

```
CalibL or CalibJ
[ ToPoint ':=' ] < expression (IN) of robtarget > ','
[ Speed ':=' ] < expression (IN) of speeddata > ','
[ GunNo ':='] < expression (IN) of num >
[\TipChg] | [\ToolChg] | [\TipWear]
[ '\' RetTipWear ':=' < variable or persistent(INOUT) of num > ]
[ '\' RetPosAdj ':=' < variable or persistent(INOUT) of num > ]
[ '\' PrePos ':=' < variable or persistent(IN) of num > ] ','
[ Zone ':=' ] < expression (IN) of zoneddata > ','
[ Tool ':=' ] < persistent (PERS) of tooldata >
[ '\' WObj ':=' < persistent (PERS) of wobjdata > ] ';'
```

Related information

	Described in:
Overview Spot Servo	*Summary* on page 9
Servo gun introduction	*Motion control* on page 191
Calibration without movement	*Motion control* on page 191
Software Equalizing	*Software Equalizing* on page 41
Software Equalizing	*Setup data for Software Equalizing* on page 58

4.1.5 Calibrate - Calibrate the gun

Description

`Calibrate` is used in spot welding to calibrate the distance between the gun tips for servo guns. This is necessary after tip change or tool change and it is recommended after welding of a number of spots or performing a tip dress. Calibrate will also update the tip wear data in the used gundata. NB The gun performs two non-synchronized close/open movements during the calibration. The open distance after the calibration is finish will be the same as before the calibration started.

If the option *Spot Servo Equalizing* is installed there are additional error handling included for lost tips when a TipChg calibration is done and supervision of the tip wear when a Tip-Wear calibration is done.

Example

```
Calibrate  gun1\ TipChange;
```

- The gun gun1 is calibrated after TipChange.
- The parameter gun1 is a num corresponding to the used gun equipment. All gun equipment used are defined in the gundata array curr_gundata in SWUSER.
- The data curr_tip_wear in curr_gundata will be automatically set to zero.

Arguments

Calibrate GunNo [\TipChg] | [\ToolChg] | [\TipWear] [\RetTipWear] | [\RetPosAdj] | [\PrePos]

GunNo **Data type: num**

Used gun equipment number. Corresponding to the element number in the gundata array curr_gundata in SWUSER

[\TipChg] **Data type: switch**

Calibration type. This calibration type is used after tip change.
The gun will close and open two times. The first close movement will be slow to find the unknown contact position. The total tip wear is reset to zero.
If the option *Spot Servo Equalizing* is selected the difference since last calibration will be

supervised. If the difference since the last calibration exceeds the supervision value in the *tipchg_superv* an error will be raised.
See *Setup data for Software Equalizing* on page 58.

[\ToolChg] **Data type: switch**

Calibration type. This calibration type is used after tool change.
The gun will close and open two times. The first close movement will be slow to find the unknown contact position. The total tip wear will remain unchanged.

[\TipWear] **Data type: switch**

Calibration type. This calibration type is used to update the tip wear and adjust the contact position after tip dress or after welding a number of spots.
The gun will close and open fast two times. The total tip wear is updated.
If the option *Spot Servo Equalizing* is selected the difference since last calibration will be supervised. If the difference since the last calibration exceeds the supervision value in the *tipwear_superv* an error will be raised.
See *Setup data for Software Equalizing* on page 58.

[\RetTipWear] **Data type: num**

The achieved tip wear [mm].

[\RetPosAdj] **Data type: num**

The positional adjustment since the last calibration [mm].

[\PrePos] **Data type: num**

The position to move with high speed to before search for contact position with slower speed is started [mm].

Program execution

Internal sequence when a Calibrate instruction is executed:

- The gun will close and open two times during the robot movement. Different tip speeds depending on selected calibration type.
- The gun is opened to the previous position.
- For certain calibration types: curr_tip_wear in the array curr_gundata in SWUSER is updated.

Positional adjustment

The optional argument RetPosAdj can be used to detect if for example the tips are lost after a tip change. The parameter will hold the value of the positional adjustment since the last calibration. The value can be negative or positive.

If the option *Spot Servo Equalizing* is selected this value will be used to calculate the difference since last calibration and supervise the tips when calibrating.

Using a pre position

In order to speed up the calibration, it is possible to define a pre position.When the calibration starts, the gun arm will be run fast to the pre position, stop and then continue slowly forward in order to detect the tip contact position. A pre position will be ignored if it is larger than the current gun position (in order not to slow down the calibration).

Error handling

Instruction parameter supervision:

The error occurs when Calibrate is called with faulty parameters or if no calibration type switch is programmed. The program stops with error text.

The parameter must be changed. When the program is restarted the current instruction is restarted from the beginning.

Tip change supervision:

If the calculated difference to the last calibration of the gun exceeds the supervision value defined in the setup data *tipchg_superv* an error will be raised and the program execution will be stopped. This error can occur for example after tip change and when Calibrate ... \TipChg is called with wrong (too large or too small tips) tips. The program stops with error message.

This error handling only exists for the *Spot Servo Equalizing option.*

Tip wear supervision:

If the calculated difference to the last calibration of the gun exceeds the supervision value defined in the setup data *tipwear_superv* an error will be raised and the program execution will be stopped. This error can occur for example after tip dressing when Calibrate ... \TipWear is called with badly dressed tips. The program stops with error message.

This error handling only exists for the *Spot Servo Equalizing option.*

Syntax

```
Calibrate
[ GunNo ':='] < expression (IN) of num >
[\TipChg] | [\ToolChg] | [\TipWear]
[ '\' RetTipWear ':=' < variable or persistent(INOUT) of num > ]
[ '\' RetPosAdj ':=' < variable or persistent(INOUT) of num > ]
[ '\' PrePos ':=' < variable or persistent(IN) of num > ];
```

Related information

	Described in:
Overview Spot Servo	*Summary* on page 9
Servo gun introduction	*Motion control* on page 191
Calibration with movement	*Motion control* on page 191
Software Equalizing	*Software Equalizing* on page 41
Software Equalizing	*Setup data for Software Equalizing* on page 58

4.1.6 OpenHighLift/CloseHighLift

Description

OpenHighLift is used in spot welding to open the gun to the highlift position (large gap).
CloseHighLift is used in spot welding to close the gun to the workstroke position (small gap).

Example

```
OpenHighLift, gun1;
```

• The gun gun1 is opened to the highlift position.

```
CloseHighLift, gun1;
```

• The gun gun1 is closed to the workstroke position.

The parameter gun1 is a num corresponding to the used gun equipment. All gun equipment used are defined in the *gundata* array curr_gundata in SWUSER.

Arguments

OpenHighLift GunNo

CloseHighLift GunNo

GunNo **Data type: *num***

Used gun equipment number. Corresponding to the element number in the gundata array curr_gundata in SWUSER

Program execution

Internal sequence when a OpenHighLift instruction is executed:

• The inhibit_close is simulated.

• The user routine SwInitUserIO is executed.

• The user routine SwOpenGun is executed and the gun is opened to the highlift position.

Internal sequence when a CloseHighLift instruction is executed:

- The inhibit_close is simulated.
- The user routine SwInitUserIO is executed.
- The user routine SwCloseGun is executed and the gun is closed to the workstroke position.

Instruction by instruction execution

Forwards

As during continuous execution.

Backwards

As during continuous execution.

Error handling

No error handling.

Syntax

```
OpenHighLift or CloseHighLift
     [ GunNo ':='] < expression (IN) of num > ';'
```

Related information

	Described in:
Overview Spot	*Summary* on page 9

4.1.7 IndGunMove - Activates independent mode for a servo gun

Description

`IndGunMove` (Independent Gun Movement) is used to set the gun in independent mode and thereafter move the gun to a specified independent position. The gun will stay in independent mode until the instruction `IndGunMoveReset` is executed.

During independent mode, the control of the servo gun is separated from the robot. The gun can be closed, opened, calibrated or moved to a new independent position, but it will not follow coordinated robot movements.

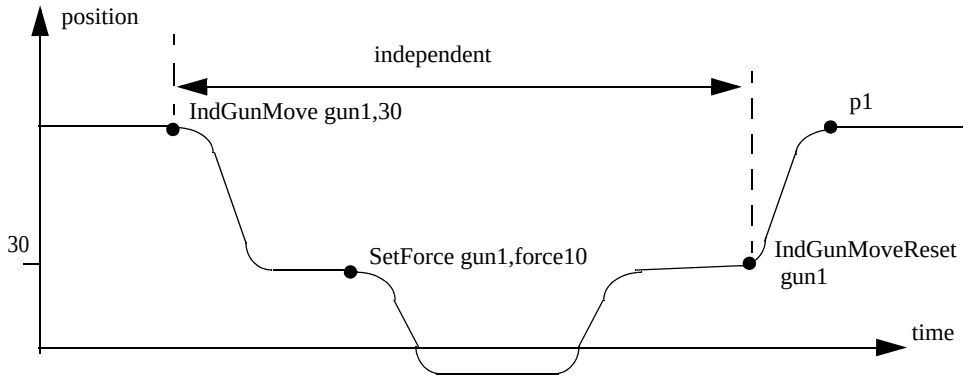
It is also possible to set the gun in independent mode from a background task while the robot in the main task can continue with for example move instructions.

For more information of how to set the gun in independent mode, see *Technical reference manual - RAPID Instructions, functions and data types - STIndGun\STIndGunReset*.

Example

```
PROC tipdress()
! Note that the gun will move to current robtarget position,
! if already in independent mode.
IndGunMoveReset gun1;
.......
.......
.......
IndGunMove gun1, 30;
......
SetForce gun1, force10;
......
IndGunMoveReset gun1;
ENDPROC
```

Independent mode is activated and the gun is moved to an independent position (30 mm). During independent mode the instruction `SetForce` is executed, without interfering with robot motion. The instruction `IndGunMoveReset` will take the gun out of independent mode and move the gun to current robtarget position.



The position p1 depends on the position of the gun given in the robtarget just performed by the robot.

Arguments

IndGunMove GunNo GunPos

GunNo **Data type: *num***

Used gun equipment number. Corresponding to the element number in the *gundata* array *curr_gundata* in SWUSER

GunPos **Data type: *num***

The position (stroke) of the servo gun in mm.

Program execution

The instruction activates independent mode and moves the gun from the coordinated position to a specified independent position. During the independent mode the gun may be closed, opened, calibrated or moved to a new independent position without interfering with robot motion.

Program restart during independent mode will always start with a regain movement to the current independent position.

The gun will recover independent mode after a system restart. Moving the program pointer will NOT reset independent mode. When the program is started, no regain movement will occur but the gun will return to independent mode after the first gun closing or calibration.

Limitations

It is not recommended that this instruction is used in combination with Spot Servo Equalizing. This will cause motion errors.

Syntax

```
IndGunMove
[GunNo ':=' <expression (IN) of num> ]
[GunPos ':=' <expression (IN) of num> ] ';'
```

Related information

	Described in:
SetForce	*SetForce - Close the gun with desired force* on page 104
STIndGun	*Technical reference manual - RAPID Instructions, functions and data types - STIndGun*

4.1.8 IndGunMoveReset - Resets servo gun from independent mode

Description

`IndGunMoveReset` (Independent Gun Movement Reset) is used to reset the gun from independent mode and thereafter move the gun to current robtarget position.

Example

```
IndGunMoveReset gun1;
```

Arguments

IndGunMoveReset GunNo

GunNo **Data type: *num***

Used gun equipment number. Corresponding to the element number in the gundata array curr_gundata in SWUSER The gun was previously set independent with the instruction *IndGunMove*.

Program execution

The instruction will reset the gun from independent mode and move the gun to current robtarget position. During this movement the coordinated speed of the gun must be zero, otherwise an error will occur. The coordinated speed will be zero if the robot is standing still or if the current robot movement includes a “zero movement” of the gun.

Syntax

```
IndGunMove
  [GunNo ':=' <expression (IN) of num> ] ';'
```

Related information

	Described in:
Definition of gundata	*gundata - Equipment specific weld data* on page 138.

4.1.9 MeasureWearL - Measure current electrode wear

Description

`MeasureWearL` is used in spot welding to measure current electrode wear for the tip on the fixed electrode. This can be done without any external measurement equipment and without manual interaction. The TCP is automatically recalculated after the measurement. The instruction also updates tip wear data in the used gundata.

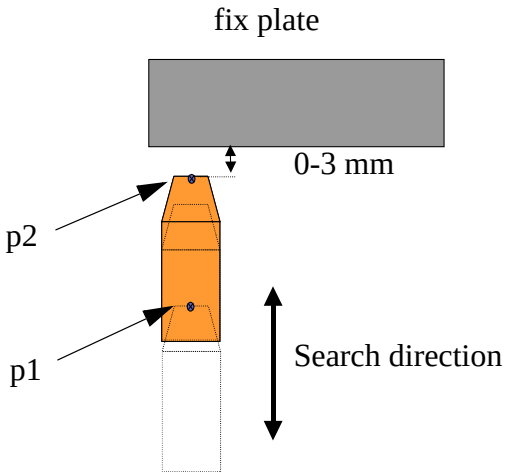
When the gun is hold by the robot the gun performs a search movement during the measurement. The gun is moving in the z direction, in the tool coordinate system, until the fixed electrode touches a fixed reference plate.

This instruction can be used also for stationary guns. In this case the robot moves the gripper and the work object until a reference position on the gripper is touching the tip on the fixed electrode.

This instruction is only available if the *Spot Servo Equalizing* option is installed.

Example

In this example the gun is held by the robot. The principles are the same also when stationary guns are used.



Measurement preparation

The measurement instruction is executed with a reference tip with an accurate TCP (tooldata in this example: *ref_tool1*). This reference measurement has to be done before the tip wear measuring is performed the first time. Also each time when the TCP for this gun is changed for some reason or if the reference plate is dislocated for some reason.

To verify if the selected measuring position or gun orientation is good enough a service routine is available; *ManCheckMeasPos*. Just run the service routine when the robot is in the selected position and you will get status information on the display.

Running the instruction `Calibrate\TipChg` after the `MeasureWearL` will also reset the total wear of the tips *curr_tip_wear* in curr_gundata and check the difference since the last calibration.

Program example for reference measurement:

```
MoveJ p1,v1000,z50,ref_tool1;
MeasureWearL p2,v1000,gun1\Reference,ref_tool1;
tool1 := ref_tool1;
! tool1 is then used during the production.
MoveL p1,v1000,z50,tool1;
Calibrate,gun1\TipChg;
```

When the `MeasureWearL` instruction with the optional argument `\Reference` is executed, first a linear movement to a position about 10 mm outside *p2* is done. Then the gun is moved in the z direction in the tool coordinate system until the fixed tip touches the reference plate. During this reference measurement the reference plate is touched twice.When the measurement is ready some reference data is stored, `tw_ref_tool` and `tw_ref_dist` in SWUSER.

The parameter `gun1` is a `num` corresponding to the used gun equipment. All gun equipment used are defined in the `gundata` array *curr_gundata* in SWUSER.

Measurement after tip wear

When it is time to compensate for current tip wear, probably after each tip dressing, the following instruction sequence should be executed:

Program example for tipwear measurement:

```
MoveJ p1,v1000,z50,tool1;
MeasureWearL p2,v1000,gun1\TipWear,tool1;
MoveL p1,v1000,z50,tool1;
Calibrate,gun1\TipWear;
```

When the instruction with the optional argument `TipWear` is executed, a search movement to the reference plate is performed and the tip wear of the fixed electrode is measured. The TCP in the used tooldata *tool1* is then recalculated and the data `curr_wear_fix` in *curr_gundata* is automatically updated.

Running the instruction `Calibrate\TipWear` after the `MeasureWearL` will also update the total wear of the tips `curr_tip_wear` in *curr_gundata* and check the difference since the last calibration.

Measurement after tip change (with or without tip dressing):

In the first measurement after tip change a similar sequence can be used as after tip wear. In this case the optional argument `\TipChange` has to be used.

Program example for tip change measurement:

```
MoveJ p1,v1000,z50,tool1;
MeasureWearL p2,v1000,gun1\TipChange,tool1;
MoveL p1,v1000,z5,tool1;
Calibrate,gun1\TipChg;
```

When the instruction with the optional argument `TipChange` is executed, similar movements as above are performed and the tip wear of the fixed electrode is measured. The TCP in the used tooldata *tool1* is then recalculated and the data `curr_wear_fix` in *curr_gundata* is automatically updated. This is the same functionality as after tip wear above. Only some extra error handling is done internally.

Running the instruction `Calibrate\TipChg` after the `MeasureWearL` will also reset the total wear of the tips `curr_tip_wear` in *curr_gundata* and check the difference since the last calibration.

Note!

It is important that p2 is the same position in all cases above. If this position is modified a new reference or reference changed measurement has to be done.

Measurement after reference plate changed

This mode can be used when the TCP for this gun is changed for some reason or if the reference plate is dislocated for some reason.

To verify if the selected measuring position or gun orientation is good enough a service routine is available; *ManCheckMeasPos*. Just run the service routine when the robot is in the selected position and you will get status information on the display.

See *Service routines (Manual Actions)* on page 164.

Program example for reference changed measurement:

```
MoveJ p1,v1000,z50,ref_tool1;
MeasureWearL p2,v1000,gun1\RefChange,ref_tool1;
MoveL p1,v1000,z50,tool1;
```

When the `MeasureWearL` instruction with the optional argument `\RefChange` is executed, first a linear movement to a position about 10 mm outside *p2* is done. Then the gun is moved in the z direction in the tool coordinate system until the fixed tip touches the reference plate. During this reference measurement the reference plate is touched twice.When the measurement is ready some reference data is stored, `tw_ref_dist` in SWUSER.

The parameter `gun1` is a `num` corresponding to the used gun equipment. All gun equipment used are defined in the `gundata` array *curr_gundata* in SWUSER.

Arguments

MeasureWearL ToPoint Speed GunNo [\Reference] | [\TipWear] | [\TipChange] | [\RefChange] Tool [\WObj]

ToPoint **Data type: *robtarget***

The destination point for the robot and additional axes. This position should be a point close to the reference position, see figure in the example above. If this position is modified a new reference measurement has to be done.

Speed **Data type: *speeddata***

The speed data that applies to movements. Speed data defines the velocity for the tool center point, the tool reorientation and additional axes.

GunNo **Data type: *num***

Used gun equipment number. Corresponding to the element number in the gundata array curr_gundata in SWUSER

[\Reference] **Data type: *switch***

Measurement type. This calibration type is used for the reference measurement with a reference tip with a well known TCP.

This measurement has to be done before the tip wear measuring is done the first time and each time when the TCP for this gun (with the reference tip mounted) is changed. It has also to be done if the reference plate (or reference position when a stationary gun is used) is dislocated of any reason.

If the reference plate is moved the switch `\RefChange` can be used instead.

[\TipWear] **Data type: *switch***

Measurement type. This measurement type is used when it is time to compensate for current tip wear, probably after each tip dressing. The data `curr_wear_fix` in *curr_gundata* will be automatically updated and the TCP in the used tooldata is recalculated.

[\TipChange] **Data type: *switch***

Measurement type. This measurement type is used in the first measurement after tip change. The data `curr_wear_fix` in *curr_gundata* will be automatically updated and the TCP in the used tooldata is recalculated.

[\RefChange] **Data type: *switch***

Measurement type. This calibration type is used if the reference plate (or reference position when a stationary gun is used) is dislocated of any reason.

The reference tool `tw_ref_tool` in swuser.sys will not be updated if this calibration type is used.

Tool **Data type: *tooldata***

The tool in use when the robot moves. The tool center point (TCP) is the point moved to the specified destination position, and should for a spot weld gun be the position on the tip of the fixed electrode.

Note!

The TCP is automatically recalculated and changed when the optional argument `\TipWear` or `\TipChange` is used.

[\WObj] **Data type: *wobjdata***

The work object (coordinate system) to which the robot position in the instruction is related.

This argument can be omitted, and if it is, the position is related to the world coordinate system. If, on the other hand, a stationary gun is used, this argument must be specified in order to perform a linear movement relative to the work object.

Program execution

Internal sequence when a `MeasureWearL` instruction is executed:

- The robot starts the movement to the destination position.
- When the destination position is reached the search movements to the reference position is started.
- The tip is touching the reference position with a predefined pressure, this force can be modified by changing the setup data *m_touch_force* in the user module *swdefusr.sys.* See section *Setup data for Software Equalizing* on page 58.
- If the optional argument \Reference is used: Some reference data is stored in the user module swuser.sys, `tw_ref_tool` and `tw_ref_dist`.
- If the optional argument \TipWear or \TipChange is used: The TCP in the used tooldata is recalculated and the data `curr_wear_fix` in *curr_gundata* is updated.

Instruction by instruction execution

Forward

As during continuous execution.

Backward

The motion is performed backwards to the destination position, but no measurement is activated.

Error handling

Following error situations are handled:

- If the search distance after tip wear measurement or measurement after tip change differs a lot from expected (for example missed tip). It is possible to change the limit values, see *Setup data for Software Equalizing* on page 58.
- If the search sequence is interrupted by for example a Stop or Emergency Stop then the search sequence is automatically restarted from the beginning at program restart.

Limitations

About how to place the fixed reference plate:

The reference plate can be mounted in an optional position in the work range, but it is necessary to orient the tool in the measuring position in that way that an **additional torque** is generated on at least one of the robot motors when the robot is touching the reference position, preferably **axis 4 to 6**.

Note!

For some special configurations the `MeasureWearL` is less suitable, for example very large guns and/or when an acceptable touch up position is not possible to reach for some reason. Then the `ReCalcTcp` method should be used instead.

Tip!

To verify if the selected measuring position or gun orientation is good enough a service routine is available; *ManCheckMeasPos*. Just run the service routine when the robot is in the selected position and you will get status information on the display. See *Service routines (Manual Actions)* on page 164.

Syntax

MeasureWearL

```
[ ToPoint ':=' ] < expression (IN) of robtarget > ','
[ Speed ':=' ] < expression (IN) of speeddata > ','
[ GunNo ':=' ] < expression (IN) of num >
[\Reference] | [\TipWear] | [\TipChange] | [\RefChange]
[ Tool ':=' ] < persistent (PERS) of tooldata >
[ '\' WObj ':=' < persistent (PERS) of wobjdata > ] ';'
```

Related information

	Described in:
System module - SWUSER	*Data* on page 154
System module - SWDEFINE	*Service routines (Manual Actions)* on page 164
Software Equalizing	*Setup data for Software Equalizing* on page 58
Overview Spot options	*Summary* on page 9

4.1.10 ReCalcTCP - Recalculate the TCP

Description

`ReCalcTCP` is used in spot welding to calculate current electrode wear for the tip on the fixed electrode and then recalculate the used TCP to compensate for current tip wear. The calculation is based on stored information about the total tip wear and about the expected tip wear ratio, the wear of the fixed tip related to the total tip wear. The instruction also updates tip wear data in the used gundata.

This instruction can be used also for stationary guns.

This instruction is only available if the *Spot Servo Equalizing* option is installed.

Example

In this example the gun can be hold by the robot or stationary. The principles are the same also when stationary guns are used.

Preparation

First the expected relation between the tip wear of the fixed tip and the total tip wear must be established, the data *tip_wear_ratio* in current gundata must be set to a relevant value. For example 50, the wear of the fixed tip is 50% of the total wear. See *gundata - Equipment specific weld data* on page 138.

This instruction has to be executed with the `\Reference` switch activated before it is used for tip wear compensation the first time.This has also to be done each time when the TCP for this gun, with new tips mounted, is changed for some reason. The TCP in the tooldata parameter, ref_tool1 in this example, has to be valid for gun1 with new tips mounted.

```
ReCalcTCP gun1\Reference,ref_tool1;
tool1 := ref_tool1;
! tool1 is then used during the production.
```

When the `ReCalcTCP` instruction with the optional argument `\Reference` is executed some reference data (the `tooldata ref_tool1`) is stored internally in the user module swuser.sys. See *Data* on page 154.

The parameter `gun1` is a `num` corresponding to the used gun equipment. All gun equipment used are defined in the `gundata` array *curr_gundata* in SWUSER.

Compensation after tip wear:

When it is time to compensate for current tip wear, probably after each tip dressing, the `ReCalcTCP` instruction is executed with the `\TipWear` switch activated. This has to be done after the gun calibration, since the total tip wear is updated during the calibration.

```
Calibrate gun1\TipWear;(CalibL/J can also be used)
ReCalcTCP gun1\TipWear,tool1;
```

When the `ReCalcTCP` instruction with the optional argument `\TipWear` is executed the TCP in the used tooldata (*tool1* in this example) is recalculated and the data *curr_wear_fix* in *curr_gundata* is automatically updated. The data `curr_tip_wear` and `tip_wear_ratio` in current `gundata` is used for the calculations.

Reset the TCP after tip change:

After tip change the instruction has to be executed, with the `\TipChange` switch activated.

```
Calibrate gun1\TipChange;
ReCalcTCP gun1\TipChange,tool1;
```

When the `ReCalcTCP` instruction with the optional argument `\TipChange` is executed, the TCP in the used tooldata, *tool1*, is set to the value used for new tips and the data `curr_wear_fix` in *curr_gundata* is cleared.

Arguments

ReCalcTCP GunNo [\Reference] | [\TipWear] | [\TipChange] Tool

GunNo **Data type: *num***

Used gun equipment number. Corresponding to the element number in the `gundata` array *curr_gundata* in SWUSER

[\Reference] **Data type: *switch***

This switch is used for preparation of the calculations.This preparation has to be done before the tip wear compensation is done the first time and also each time the TCP for this gun (with new tips mounted) is changed.

The TCP in tooldata has to be valid for a gun with new tips mounted.

[\TipWear] **Data type: *switch***

This switch is used when it is time to compensate for current tip wear, probably after the gun calibration after each tip dressing. The data `curr_wear_fix` in *curr_gundata* will be automatically updated and the TCP in the used tooldata is recalculated.

[\TipChange] **Data type: *switch***

This switch is used when the instruction is executed after tip change. The data `curr_wear_fix` in *curr_gundata* is cleared and the TCP in the used tooldata is set to the value valid for new tips.

Tool **Data type: *tooldata***

Tooldata for the used gun. The tool center point (TCP) should for a spot weld gun be the tip position for the fixed electrode tip.

Note!

The TCP in current `tooldata` is automatically recalculated and changed when the optional argument `\TipWear` or `\TipChange` is used.

Program execution

Internal sequence when a `ReCalcTCP` instruction is executed:

- If the optional argument `\Reference` is used: Some reference data is stored internally.
- If the optional argument `\TipWear` is used: The TCP in the used tooldata is recalculated and the data `curr_wear_fix` in *curr_gundata* is updated.
- If the optional argument `\TipChange` is used: The TCP in the used tooldata is set to a value valid for new tips and the data `curr_wear_fix` in *curr_gundata* is cleared.

Error handling

Following error situations are handled:

- If the calculated tip wear differ a lot from expected (for example missed tip). It is possible to change the limit values, see *Setup data for Software Equalizing* on page 58.

Syntax

```
ReCalcTCP
   [ GunNo ':='] < expression (IN) of num >
   [\Reference] | [\TipWear] | [\TipChange]
   [ Tool ':=' (PERS) of tooldata >';'
```

Related information

	Described in
System Module SWUSER	*Data* on page 154
System Module SWDEFUSR	*Data definitions* on page 166
Gundata	*gundata - Equipment specific weld data* on page 138
Software Equalizing	*Software Equalizing* on page 41
Overview Spot options	*Summary* on page 9

4.2 Data types

4.2.1 forcedata - Spot gun force data

Description

`Forcedata` is used to define the parameters for control of the spot weld gun when it is closed without welding (with a `SetForce` instruction or with manual actions).

`Forcedata` has the following default structure when servo guns are used:

- Desired gun tip force.
- Desired force time.
- Expected total plate thickness.(Only valid for servo guns).

Allowed variation when checking the plate thickness.(Only valid for servo guns).

Components

tip_force (gun tip force) Data type: num

Defines the desired gun tip force. [N] If pneumatic gun, this value controls a group output.

force_time (force time) Data type: num

Defines the desired gun force time [s]

plate_thickness (plate thickness) Data type: num

Defines the expected total plate thickness. [mm] (Only valid for servo guns).

plate_tolerance (plate tolerance) Data type: num

Defines the allowed variation when checking the plate thickness [mm]
If the value is 0 the thickness check is deactivated. (Only valid for servo guns).

Predefined data

Servo guns:

```
PERS forcedata force1 := [1000, 1, 0, 0];
```

Pneumatic guns:

```
PERS forcedata force1 := [1, 1];
```

Defined in module SWDEFUSR.

force1 is used as default in the first programmed `SetForce` instruction and has following default data:.

- Desired gun tip force = 1000 N.(Gun pressure 1 for pneumatic gun).
- Desired force time = 1 s.
- Expected total plate thickness = 0 mm.(Servo gun).
- Allowed variation in the thickness = 0 (thickness check is deactivated) (Servo gun).

Servo guns:

```
PERS forcedata curr_forcedata{2} := [[0,0,0,0],[0,0,0,0]];
```

Pneumatic guns:

```
PERS forcedata curr_forcedata{2} := [[0,0],[0,0]];
```

Defined in module SWUSER.

curr_forcedata is an array with active or latest used forcedata parameters for each defined gun. This parameters are automatically updated by the kernel when a `SetForce` instruction is executed. The parameters are used when gun closure is manually activated, see *Manual actions* on page 29.

Customizing

The Spot package provides opportunities for the user to customize the functionality to adapt to different types of spot weld equipment and user defined standards. For this data type it is possible to delete components if they are not used. It is also possible to give the components own user defined names.

However, the main subject of this description is the default setup. See *Customizing* on page 237.

Default structure

For servo guns:

```
<dataobject of forcedata>
    <tip_force of num>
    <force_time of num>
    <plate_thickness of num>
    <plate_tolerance of num>
```

For pneumatic guns:

```
<dataobject of forcedata>
    <tip_force of num>
    <force_time of num>
```

Related information

	Described in:
The SetForce instruction	*SetForce - Close the gun with desired force* on page 104
Overview Spot & Spot Servo	*Summary* on page 9
Customizing possibilities	*Customizing* on page 237
Definition of spotdata	*spotdata - Spot weld data* on page 148
System module SWUSER	*System modules* on page 153

4.2.2 gundata - Equipment specific weld data

Description

`gundata` is used to define spot weld equipment specific data, to control the gun in an optimal way in the weld process when the spot instructions are used. Each gundata defines one gun equipment.

Note!

The `gundata` structure differs between different spot options.

Description

`Gundata` has the following default structure when servo guns are used:

- Gun name
- Closing and equalizing times.
- Weld counters and a max value.
- Current tip wear and a max value.
- The weld timeout time.
- Gun data for the Software Equalizing functions. (Only if Spot Servo Equalizing is installed.)

`Gundata` has the following default structure when pneumatic guns are used:

- Gun name
- Closing and equalizing times.
- Opening and equalizing times.
- Weld counters and a max value.
- The close timeout time.
- The weld timeout time.
- The open timeout time.

Components

gun_name (gun name) Data type: string

The name of the mechanical unit used for the servo gun. This name must be identical with the name of the mechanical unit defined in the motion servo gun parameters.

pre_close_time (preclose time) Data type: num

Time [s] before gun in weld position, when the asynchronous gun closure is started. For pneumatic guns, when the gun close signal is set.

This data is not used if Software Equalizing is active. In this case the preclosing is handled automatically during the movement from the release distance to the weld position.

pre_equ_time (pre-equalize time) Data type: num

Time [s] before gun in weld position, when a digital output signal is set for activation of a mechanical gun equalizing system in the gun, if used.

open_time (open time) Data type: num

Time [s] that elapses between the gun opening order and the release of the next motion or the the test of gun open. (Pneumatic guns only).

post_equ_time (post-equalize time) Data type: num

Time [s] after the gun close signal has been reset (gun open), when the gun equalizing is deactivated. (Pneumatic guns only).

weld_counter (weld counter) Data type: num

Counter for the number of welds done with this gun. The counter is automatically incremented. Use of this data is optional. Zero set shall be handled by the user program.

max_nof_welds (max number of welds) Data type: num

Max number of performed welds. Use of this data is optional.

curr_tip_wear (current tip wear) Data type: num

Current tip wear [mm]. This data is automatically updated after each gun calibration. Use of this data is optional. (Servo guns only).

max_tip_wear (max tip wear) Data type: num

Max allowed tip wear before tip exchange [mm]. Use of this data is optional. (Servo guns only).

close_timeout (close timeout) Data type: num

The max time[s] waiting for gun open signal before closing the gun, after this time the error handling is activated. (Pneumatic guns only).

weld_timeout (weld timeout) Data type: num

The max time[s] waiting for weld_complete before the error handling is activated.

open_timeout (open timeout) Data type: num

The max time[s] waiting for gun open signal after the gun has been opened, after this time the error handling is activated. (Pneumatic guns only).

If Spot Servo Equalizing is installed also following components are present:

softw_equ (software equalizing) Data type: bool

This data has to be set to TRUE to activate the software equalizing functions Release of the fixed gun arm and Deflection Compensation.

curr_wear_fix (current tip wear for the fixed tip) Data type: num

Current tip wear for the fixed gun electrode tip [mm]. This data is automatically updated when MeasureWearL is used

wear_moveable (current tip wear for the moveable tip) Data type: num

Current tip wear for the moveable gun electrode tip [mm]. This data is automatically updated when CalibL/J and Calibrate is used.

release_dist (release distance) Data type: num

The release distance [mm] when the robot is moving between weld positions during normal program execution and during Weld position Touch Up.

deflection_dist (deflection distance) Data type: num

TCP deviation [mm] caused of gun arm deflection when the gun is closed with the force deflection_force. This data is used for the deflection compensation.

deflection_force (deflection force) **Data type: num**

Applied force [N] corresponding to the TCP deviation deflection_dist caused of gun arm deflection. This data is used for the deflection compensation.

deflection_time (deflection time) **Data type: num**

The time for the gun to build up the gun force [s]. This data is used for the deflection compensation. If no data information exists, use the default value (0.1 s).

tip_wear_ratio (tip_wear_ratio) **Data type: num**

The expected ratio [%] between the tip wear for the fix tip related to the total tip wear. This value has to be set to a permitted value (0-100) if the calculation method (ReCalcTCP) is used for the tip wear compensation. If the measuring method is used (MeasureWearL) this value has to be set to -1.

Default structure

For servo guns if not Spot Servo Equalizing is installed:

```
<dataobject of gundata>
    <gun_name of num>
    <pre_close_time of num>
    <pre_equ_time of num>
    <weld_counter of num>
    <max_nof_welds of num>
    <curr_tip_wear of num>
    <max_tip_wear of num>
    <weld_timeout of num>
```

For servo guns if Spot Servo Equalizing is installed:

```
<dataobject of gundata>
    <gun_name of num>
    <pre_close_time of num>
    <pre_equ_time of num>
    <weld_counter of num>
    <max_nof_welds of num>
    <curr_tip_wear of num>
```

```
    <max_tip_wear of num>
    <weld_timeout of num>
    <softw_equ of bool>
    <curr_wear_fix of num>
    <wear_moveable of num>
    <release_dist of num>
    <deflection_dist of num>
    <deflection_force of num>
    <deflection_time of num>
    <tip_wear_ratio of num>
```

For pneumatic guns:

```
< dataobject of gundata>
    <gun_name of num>
    <pre_close_time of num>
    <pre_equ_time of num>
    <open_time of num>
    <post_equ_time of num>
    <weld_counter of num>
    <max_nof_welds of num>
    <close_timeout of num>
    <weld_timeout of num>
    <open_timeout of num>
```

Predefined data

For servo guns if not Spot Servo Equalizing is installed:

```
PERS gundata curr_gundata{4} :=
  [[“SERVOGUN”, 0.1, 0, 0, 1000, 0, 10, 2],
   [“NOT USED”, 0.1, 0, 0, 1000, 0, 10, 2],
   [“NOT USED”, 0.1, 0, 0, 1000, 0, 10, 2],
   [“NOT USED”, 0.1, 0, 0, 1000, 0, 10, 2]];
```

For servo guns if Spot Servo Equalizing is installed:

```
PERS gundata curr_gundata{4} :=
  [[“SERVOGUN”, 0.05, 0, 0, 1000, 0, 10, 2, TRUE, 0, 0, 10,
0, 5000,0.1,-1],
   [“NOT USED”, 0.05, 0, 0, 1000, 0, 10, 2, TRUE, 0, 0, 10,
0, 5000,0.1,-1],
   [“NOT USED”, 0.05, 0, 0, 1000, 0, 10, 2, TRUE, 0, 0, 10,
0, 5000,0.1,-1],
   [“NOT USED”, 0.05, 0, 0, 1000, 0, 10, 2, TRUE, 0, 0, 10,
0, 5000,0.1,-1]];
```

For pneumatic guns:

```
PERS gundata curr_gundata{4} :=
  [["PNEU_G1", 0.05, 0.02, 0, 0, 0, 10000, 2, 2, 2],
   ["PNEU_G2", 0.05, 0.02, 0, 0, 0, 10000, 2, 2, 2],
   ["PNEU_G3", 0.05, 0.02, 0, 0, 0, 10000, 2, 2, 2],
   ["PNEU_G4", 0.05, 0.02, 0, 0, 0, 10000, 2, 2, 2]];
```

curr_gundata is an array with active gundata parameters for each used gun. These parameters have to be changed by the user during the installation and programming phase to be in agreement with the weld equipment in use. In the default package, *curr_gundata* is defined in module SWUSER.

Customizing

The Spot package provides opportunities for the user to customize the functionality to adapt to different types of spot weld equipment and user defined standards. For this data type it is possible to delete components if they are not used. It is also possible to give the components own user defined names.

However, the main subject of this description is the default setup.

See *Customizing* on page 237.

.**Related information**

	Described in:
SpotL/J	*SpotL/SpotJ - The basic spot welding instructions* on page 73
Overview Spot Options	*Summary* on page 9
Customizing possibilities	*Customizing* on page 237
Definition of spotdata	*spotdata - Spot weld data* on page 148
System module SWUSER	*System modules* on page 153

4.2.3 simdata - Simulation data

Description

`Simdata` is used to define the parameters that control the different simulation modes used when testing spot weld programs.

`Simdata` has the following default structure when servo guns are used:

- Desired simulation type.
- Desired simulation time.
- If testing is performed with/without gun closure.
- If testing is performed with/without plates.(Only valid for servo guns).

Components

sim_type (simulation type) Data type: num

Desired simulation type. Permitted values:

0 - All simulations are deactivated.

1 - Simulation of the weld is performed in the robot controller. (No start signal)

2 - Simulation of the weld is performed in the weld controller. (enable_current = 1)

3 - Weld position Touch Up mode. Only available if Spot Servo Equalizing is installed.

sim_time (simulation time) Data type: num

Defines the desired simulation time [s] when simulation of the weld is performed in the robot controller (sim_type = 1).

inhib_close (inhib close) Data type: bool

Testing without closing the guns. Only relevant if sim_type = 1 or 2.

no_plates (no plates) Data type: bool

Testing without plates. Only relevant if sim_type = 1 or 2. (Only valid for servo guns).

Predefined data

```
Servo guns:
PERS simdata data curr_simdata := [0, 0.5, FALSE, FALSE];
Pneumatic guns:
PERS simdata data curr_simdata := [0, 0.5, FALSE];
```

Defined in module SWUSER.

curr_simdata is holding all active simulation data. This data influences all used weld equipment when `SpotL/J` or `SpotML/J` instructions are executed. The user has to change this data to activate a simulation mode. All simulations are deactivated if `sim_type = 0` (default).

Customizing

The Spot package provides opportunities for the user to customize the functionality to adapt to different types of spot weld equipment and user defined standards. For this data type it is possible to delete components if they are not used. It is also possible to give the components own user defined names. However, the main subject of this description is the default setup.
See *Customizing* on page 237.

Default structure

For servo guns:

```
<dataobject of simdata>
    <sim_type of num>
    <sim_time of num>
    <inhib_close of bool>
    <no_plates of bool>
```

For pneumatic guns:

```
<dataobject of simdata>
    <sim_type of num>
    <sim_time of num>
    <inhib_close of bool>
```

Related information

	Described in:
The SpotL instruction	*SpotL/SpotJ - The basic spot welding instructions* on page 73
The SpotML instruction	*SpotML/SpotMJ - Spot welding with multiple guns* on page 89
Overview Spot & Spot Servo	*Summary* on page 9
Customizing possibilities	*Customizing* on page 237
System module SWUSER	*System modules* on page 153

4.2.4 spotdata - Spot weld data

Description

`Spotdata` is used to define the parameters that control the weld equipment when welding a certain spot.

`Spotdata` is used by the `SpotL/J` and `SpotML/J` instructions and contains data which controls the welding of one spot.

`Spotdata` has the following default structure when servo guns are used:

- Program number for the program in the weld timer to be used.
- Desired gun tip force.
- Expected total plate thickness.(Only valid for servo guns).
- Allowed variation when checking the plate thickness.(Only valid for servo guns).

Components

prog_no (program number) Data type: num

Defines the internal program in the weld timer to be used for the welding.
Permitted values: 0 - 255 (defined in the system module SWDEFUSR).

tip_force (gun tip force) Data type: num

Defines the desired gun tip force. [N]
If pneumatic gun, this value controls a group output.

plate_thickness (plate thickness) Data type: num

Defines the expected total plate thickness. [mm]
(Only valid for servo guns).

plate_tolerance (plate tolerance) Data type: num

Defines the allowed variation when checking the plate thickness [mm]
If the value is 0 the thickness check is deactivated.
(Only valid for servo guns).

Predefined data

Servo guns:

```
PERS spotdata spot1 := [1, 1000, 0, 0];
```

Pneumatic guns:

```
PERS spotdata spot1 := [1, 1];
```

Defined in module SWDEFUSR.

Spot1 is used as default in the first programmed Spot instruction and has following default data:

- The program number 1 in the weld controller shall be used.
- Desired gun tip force = 1000 N.(Gun pressure level 1 for pneumatic guns).
- Expected total plate thickness = 0 mm.(Servo guns).
- Allowed variation in the thickness = 0 (thickness check is deactivated) (Servo guns).

Servo guns:

```
PERS spotdata
curr_spotdata{4} :=[[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0
]];
```

Pneumatic guns:

```
PERS spotdata curr_spotdata{4} :=[[0,0],[0,0],[0,0],[0,0]];
```

Defined in module SWUSER.

curr_spotdata is an array with active or latest used `spotdata` parameters for each defined gun. This parameters are automatically updated by the kernel when spot instructions are executed. This `spotdata` are used for reweld situations and if welding is manually activated (see *Manual actions* on page 29)

Customizing

The Spot package provides opportunities for the user to customize the functionality to adapt to different types of spot weld equipment and user defined standards. For this data type it is possible to delete components if they are not used. It is also possible to give the components own user defined names.

However, the main subject of this description is the default setup.

See *Customizing* on page 237.

Default structure

For servo guns:

```
<dataobject of spotdata>
    <prog_no of num>
    <tip_force of num>
    <plate_thickness of num>
    <plate_tolerance of num>
```

For pneumatic guns:

```
<dataobject of spotdata>
    <prog_no of num>
    <tip_force of num>
```

Related information

	Described in:
SpotL/J	*SpotL/SpotJ - The basic spot welding instructions* on page 73
Overview Spot Options	*Summary* on page 9
Customizing possibilities	*Customizing* on page 237
Definition of gundata	*gundata - Equipment specific weld data* on page 138
System module SWUSER	*System modules* on page 153

4.2.5 smeqdata - SoftMove Equalizing data

Description

`Smeqdata` is used to define the parameters used to control the SoftMove function that can be used when welding a certain spot. This data has to be used by the `SpotL/J` instructions if SoftMove equalizing is required.

`Smeqdata` has the following structure:

- Desired *stiffness* of the robot for the specific position.
- Desired *force_offset* for the specific position.

Components

stiffness (spring effect of the robot) Data type: num

Defines how strongly the robot tries to move back to the reference point when it is pushed away from that point. The value is a percentage of a configured value in the motion parameter configuration. Possible values for this parameter are inbetween 0 and 100%. 0 gives no spring effect of going back to the reference point.

force_offset (friction compensation) Data type: num

Defines the desired force in Newton, corresponding to the static friction of the robot in the soft z direction. If the force value is too low it will be hard to push the robot. On the other hand, if the force value is too high the robot will float away by itself. Possible values for this parameter are inbetween 0.1 and 2000 N. The friction compensation needed will be measured for each SpotL/J instruction if the initial value is zero, see *SoftMove Equalizing* on page 60.

Note!

Note that the measured *force_offset* value is only a rough reference value and in most cases it will *not* be very accurate.

For more information about SoftMove see *Application manual – SoftMove*.

Example

Program example with and without SoftMove Equalizing functionality:

```
PERS smeqdata smeq10:=[1,50];
PERS smeqdata smeq11:=[1,70];

PROC main()

  MoveJ P10, v1000, z50, tool1;
  SpotL P20, vmax, gun1, spot11, tool1;
  SpotL P30, vmax, gun1, spot12\ SMEQ:=smeq10, tool1;
  SpotL P40, vmax, gun1, spot13\ SMEQ:=smeq11, tool1;
  SpotL P50, vmax, gun1, spot14, tool1;

ENDPROC
```

Default structure

```
<dataobject of smeqdata>
  <stiffness of num>
  <force_offset of num>
```

Related information

	Described in:
SpotL/J	*SpotL/SpotJ - The basic spot welding instructions* on page 73
SoftMove Equalizing	*SoftMove Equalizing* on page 60
SoftMove	Application manual - SoftMove

5 System modules

5.1 SWUSER

5.1.1 Description

This module is intended for the person who creates and tests the user program.

SWUSER is running in all tasks and contains current values for the different defined SpotWare data types. It also contains the different process hooks (for example. SwPrepare and SwCloseGun).

Note!

The default functionality is NOT the same for all Spot options.

Tip!

See <Italic>Customizing on page 237

Note!

After changing any routines in SWUSER, the following steps must be taken before there is an effect on the application:

- Save SWUSER. The old one is overwritten.
- Generate a P-Start to affect all tasks.

5.1.2 Data

The names are predefined and used internally when Spot instructions are used. They must therefore not be deleted or renamed.

The following global data are predefined:

Note!

Some of the data only concerns Spot servo equalizing.

Note!

If data is moved from this module the Spot MMI application might not work properly!

Name	Declaration	Description
curr_gundata{4}	PERS gundata	Current gun specific data for gun equipment 1 to 4.
curr_spotdata{4}	PERS spotdata	Current or latest used spot data for gun equipment 1 to 4. Is automatically updated from the instruction before the first process hook is called. This data is used when the manual action ManSpot is activated.
curr_forcedata{4}	PERS forcedata	Current or latest used forcedata for gun equipment 1 to 4. Is automatically updated when force instructions are running.
man_forcedata{4}	PERS forcedata	Forcedata for gun equipment 1 to 4 used when manual actions are activated (ManSetForce and ManCloseGun).
curr_simdata	PERS simdata	Current parameters for simulation. These parameters have influence on all used equipment.
force_calib	PERS bool	Memory variable to supervise a tuned gun, will be set to TRUE in ManForceCalib after a completed force calibration. NOTE! If this boolean is set without performing a force calibration the servogun can be damaged when run. TRUE = gun is tuned. FALSE = gun is not tuned.

Name	Declaration	Description
tw_ref_dist{4}	PERS num	Reference Data for tip wear measurement. Distance to the reference surface for the reference fixed tip. Only for Spot Servo Equalizing.
tw_ref_tool{4}	PERS tooldata	Reference Data for tip wear measurement. Reference tooldata. Only for Spot Servo Equalizing

Following signal references are predefined. After the routine SwInitUserIO has been executed they will reference to the physical signal on the activated equipment..

Name	Declaration
close_gun	VAR signaldo
open_hlift	VAR signaldo
close_hlift	VAR signaldo
equalizing	VAR signaldo
press_group	VAR signalgo
pressure_ok	VAR signaldi
gun_open	VAR signaldi
hlift_open	VAR signaldi
reset_fault	VAR signaldo
timer_ready	VAR signaldi
start_water	VAR signaldo
weld_contactor	VAR signaldi
flow1_ok	VAR signaldi
flow2_ok	VAR signaldi
temp_ok	VAR signaldi
air_ok	VAR signaldi
equipment_ok	VAR signaldi

Note!

Some of the signals are only used if a pneumatic gun is used.

5.1.3 Process hooks

Introduction

The following predefined routines are installed with the application. They are all used by the spotweld instructions and called from the kernel during the process.

These routines have a default functionality but can easily be modified to fit specific equipments. The routines cannot be deleted or renamed since they are called from internal modules.
Parameters description for the process hooks:

- num GunNum: Gun equipment number.
- INOUT ErrText: Error message. If an error text is returned in this parameter it will generate an error dialog with possibilities for the operator to decide what to do. If ErrText = “Retry”, that is SwTextGet(3) is returned from some of the hooks then no interaction with the operator will be performed. The process is restarted from the beginning.

PROC SwInitUserIO(num GunNum)

The routine is the first called process hook, called in the beginning of the motion part.

Default functionality:

The signal names used in the hooks in this module is connected to the physical used signals. As default, signals for four gun equipment are defined.

PROC SwPrepare(num GunNum, INOUT string ErrText)

The routine is called in the beginning of the motion part but after SwInitUserIO.

Default functionality:

If no simulations are active then a number of status signals from the weld equipment are tested. If something is wrong a relevant text string is put in the ErrText parameter.

PROC SwCloseGun(num GunNum, INOUT string ErrText)

The routine is called a predefined time (pre_close_time in curr_gundata) before the TCP reaches the weld position. After leaving this hook successfully, the gun will be ordered to close.

For servogun no default functionality.

Default functionality for a pneumatic gun:

The gun will be set to workstroke position and then the gun will be closed.

PROC SwPreWeld(num GunNum, INOUT string ErrText)

This routine is called in the weld position and is the last routine to be called before the start signal to the timer is activated.

Default functionality:

If no simulations are active then the timer ready signal from the weld timer is tested. If something is wrong a relevant text string is put in the ErrText parameter.

PROC SwOpenGun(num GunNum, INOUT string ErrText)

This routine is called just after receiving the weld_complete signal, before the open gun order is activated.

Default functionality for servo gun:

If no simulations are active then the weld counter in curr_gundata is updated.

Default functionality for pneumatic gun

The gun will be opened and the gun will be set to highlift or workstroke position according to the choice in the spot instruction.

PROC SwPostWeld(num GunNum, INOUT string ErrText)

This routine is called just after the process hook SwOpenGun.

Default functionality for pneumatic gun:

If no simulations are active then the weld counter in curr_gundata is updated.

PROC SwWeldFault(num GunNum, INOUT string ErrText)

This routine is called when the weld_timeout time has elapsed without receiving the weld_complete signal, or when receiving the fault signal from the weld timer during the weld sequence. The gun has been ordered to open just before this hook is called.

No default functionality.

PROC SwErrorInfo(num GunNum, string ErrType, string ErrText)

This routine is called when the info button has been tapped in an error dialog. It is possible to add extra error information here.

Default functionality:

- The text strings in ErrType and ErrText is written on the display plus the text string “No further information”. A push button “Return” appears.
- For weld errors also the robtarget name is written (if present).

Possible ErrType values:

- SwTextGet(136)Error generated in SwPreWeld.
- SwTextGet(134)Error generated in SwCloseGun.
- SwTextGet(133)Error generated in SwPrepare.
- SwTextGet(137)Error generated in SwOpenGun.
- SwTextGet(132)Weld error.

PROC SwResetFault(num GunNum)

This routine is used to reset the weld timer after fault situations.

Default functionality:

The reset signal for current gun is pulsed.

PROC SwErrorRecover(num GunNum, string ErrType, string ErrText, num CurrThickness, INOUT num Status)

This routine is called instead of the built-in error handling if the boolean *user_recover* is set to TRUE. When using this routine it is possible to customize the error dialogs on the FlexPendant when an error has occurred. FALSE means that standard error recovery is used.

Parameters

- GunNum: Current gun number.
- ErrType: Type of error that occurred. Possible cases are:
  - SW_WELD_ERR: weld error timeout.
  - SW_CLOSE_GUN_ERR: close gun error. Error reported by SwCloseGun.
  - SW_OPEN_GUN_ERR: open gun error. Error reported by SwOpenGun.
  - SW_PREPARE_ERR: prepare error. Error reported by SwPrepare.
  - SW_PRE_WELD_ERR: preweld supervision error. Error reported by SwPreWeld.
  - SW_TIP_POS_ERR: tip position error. (Only for servo guns)
- ErrText: text string that was returned by the function that reported the error.

The return values of this function defines how the Spot options shall resume after this error.

There are three possible return values:

- SW_RETRY: the weld process is started from the beginning after weld error and after errors reported by:
  - SwPrepare
  - SwCloseGun
  - SwPreWeld
  - SwOpenGun.
- SW_SKIP: the current spot weld process is abandoned and cleaned up.
- SW_IGNORE: the current tip position error is ignored and the weld is executed again without plate thickness supervision. (Only for servo guns)

5.2 SWDEFINE

5.2.1 Description

This module is intended for the person who does advanced customizing.

SWDEFINE is running in all motion tasks and contains event routines for power on, start, stop etc.

It also contains the different service routines (Manual Actions), for example ManSpot and ManServiceCalib.

Note!

After changing any routines in SWDEFINE, the following steps must be taken before there is an affect on the application:

- Save SWDEFINE. The old one is overwritten.
- Reload the module into the motion tasks manually or perform a p-start to reload the module.

Note!

The default functionality in this module is NOT the same for all Spot options.

5.2.2 Data

The following global data is predefined:

Data	Description	Default value
forcecaldata	Contains the setup data for ManForceCalib. This data will be saved when the ManForceCalib routine is run the first time.	Number of calibrations 5, Maximum force 5000 N, Sensor thickness 20 mm, Force time 1 sec.

5.2.3 Event routines

The following predefined event routines are installed with the application. These routines have a default functionality but can easily be changed. The routines cannot be deleted since they are called from internal modules.

PROC SwPowerOn()

This routine is called when the robot is restarted (warm start) from the FlexPendant or by power on.

Default functionality:

The signal names used internally is connected to the physical used signals. By default, signals for four gun equipment are defined.

PROC SwStart()

This routine is called when execution is started from the beginning of the program.

No default functionality:

PROC SwReStart()

This routine is called when execution is started from the position where it was stopped.

No default functionality.

PROC SwStop()

This routine is called when the program is stopped.

No default functionality:

PROC SwQStop()

This routine is called when the robot is quick stopped (E-stop).

No default functionality:

5.2.4 Process routines

These routines can be used to perform actions inside the Spot routines.
The following process routines are installed with the application. These routines has no default functionality but can easily be changed. The routines can be deleted if not needed.

PROC DefineUsrData(num GunNum)

This routine is called in the beginning of each Spot shell routine.

Here for example user data can be transferred to the Spot instruction, using this data instead of the default data, see example below.

```
curr_gundata{GunNum}.gun_name := user_gundata.gun_name;
curr_gundata{GunNum}.pre_close_time := user_gundata.pre_close_time;
curr_gundata{GunNum}.pre_equ_time := user_gundata.pre_equ_time;
curr_gundata{GunNum}.weld_counter := user_gundata. weld_counter;
DefineGunData;
```

No default functionality:

PROC UpdateUsrData(num GunNum)

This routine is called at the end of each Spot shell routine.

Here for example process data can be transferred back to the user data, see example below.

```
user_gundata.weld_counter := curr_gundata{GunNum}.weld_counter;
user_gundata.curr_tip_wear := curr_gundata{GunNum}.curr_tip_wear;
```

No default functionality:

5.2.5 Service routines (Manual Actions)

The following predefined service routines are installed with the application. These routines have a default functionality but can be changed if needed.

PROC ManSpot()

This routine performs a manual weld with the latest used spotdata at the current position with the current gun.

PROC ManCalib()

This routine performs a manual gun calibration the type is selected when running this routine. Servo guns only.

PROC ManServiceCalib()

This routine performs a manual service calibration for the selected gun.
There are two options available when running this routine, synchronizing the tip position after revolution counter update or initializing the servo gun position after fine calibration. Servo guns only.

PROC ManForceCalib()

This routine performs a manual force calibration with the selected gun.
Servo Guns only.

PROC ManCloseGun()

This routine performs a manual gun close according to the values in man_forcedata.

PROC ManOpenGun()

This routine performs a manual gun open with the selected gun.

PROC ManSetForce()

This routine performs a manual gun close and gun opening according to the values in man_forcedata.

PROC ManAddGunName()

This routine searches for servoguns in the system, and gives the possibility to add the names to curr_gundata{selected gun number}.gun_name. Servo guns only.

PROC ManCloseHLift()

This routine closes the gun to the workstroke position. Pneumatic guns only.

PROC ManOpenHLift()

This routine opens the gun to the high lift position. Pneumatic guns only.

PROC ManCheckMeasPos()

This routine can be used to verify if a robot position or gun orientation is suitable for a tip wear measurement with MeasureWearL.

When this routine is run in the selected position status information will be presented on the FlexPendant whether the position is good or bad.
The recommended touch up axis should be 4 to 6 and the touch up value should be in between 0.25 and 1.

This instruction is only available if the *Spot Servo Equalizing* option is installed.

Note!

For some special configurations the MeasureWearL measuring method is less suitable, for example very large guns and/or when an acceptable touch up position is not possible to reach for some reason. Then the ReCalcTcp method should be used instead.

5.3 SWDEFUSR

5.3.1 Description

This module is intended for the person who does advanced customizing.

SWDEFUSR is running in all tasks and contains all the data definitions and setup data.

It also contains the different data definition routines (for example DefineSpotData and DefineGunData).

Note!

The default functionality is NOT the same for all Spot options.

Note!

After changing any routines in SWDEFUSR, the following steps must be taken before there is an affect on the application:

- Save SWDEFUSR. The old one is overwritten.
- Generate a P-Start to affect all tasks

Note!

If data is moved from this module the Spot MMI application might not work properly!

5.3.2 Data definitions

The following global data records are predefined.

Record data	Description	Default value
forcedata	Definition of force data.	num tip_force; num force_time; num plate_thickness; num plate_tolerance;
simdata	Definition of simulation data.	num sim_type; num sim_time; bool inhib_close; bool no_plates;

Record data	Description	Default value
spotdata	Definition of process spot data	num prog_no; num tip_force; num plate_thickness; num plate_tolerance;
gundata	Definition of process gun data.	string gun_name; num pre_close_time; num pre_equ_time; num weld_counter; num max_nof_welds; num curr_tip_wear; num max_tip_wear; num weld_timeout; bool softw_equ; num curr_wear_fix; num wear_moveable; num release_dist; num deflection_dist; num deflection_force; num deflection_time; num tip_wear_ratio;

Note!

Some of the data only concerns servo guns and Software Equalizing.

5.3.3 Setup data

The following min and maximum data and setup data are predefined. These limits will be supervised when running spot instructions.

Name	Description	Default value
MAX_TIP_FORCE	Maximum allowed tip force	6000 N
MAX_PRE_CLOSE_T	Maximum allowed preclose time of the gun	0.5 s
MAX_PRE_EQU_T	Maximum allowed pre-equalizing time for the gun	0.1 s
MAX_PLATE_THICK	Maximum allowed plate thickness	40 mm
MAX_PLATE_TOLER	Maximum allowed plate tolerance	1 mm
MAX_PROG_NO	Maximum allowed program number	255
MIN_WELD_TOUT	Minimum allowed weld timeout	2 s
MAX_WELD_TOUT	Maximum allowed weld timeout	10 s
MIN_FORCE_T	Maximum allowed force time	0.05 s
MAX_FORCE_T	Minimum allowed force time	10 s
MAX_REL_DIST	Maximum allowed release distance	15 mm
MAX_DEFLECTION	Maximum allowed deflection distance	15 mm
MIN_TOUCHUP_STEP	Minimum allowed touch up step	0.1 mm
MAX_TOUCHUP_STEP	Maximum allowed touch up step	10 mm, absolute max is 50 mm.
MAX_SIM_T	Maximum allowed simulation time	10 s

Name	Description	Default value
SW_NOF_GUNS	Number of defined guns in the system If this value is changed all indexed data must be changed accordingly	4
gun1, 4	Gun number used in the spot instruction, gun index number in curr_gundata	1 to 4
auto_reweld	Automatic reweld: Number of automatic tries to reweld after weld_complete timeout	0

Name	Description	Default value
async_start	Asynchronous start: The start of the weld is set independently of inpos, that is immediately after closing the gun	FALSE
prog_num_parity	Program parity: 0 = none, 1 = odd, 2 = even	0
manage_equalize{number of robots}	Defines if the equalize signal should be set/reset from the process kernel	TRUE
check_data	Defines if the spotdata parameters should be checked during process	TRUE
user_recover	User defined error handling. The error handling routine SwErrorRecover in swuser.sys is called if this boolean is set to TRUE	FALSE
spot1	Default spotdata when programming spot instructions on the FlexPendant	prog_no - 1 tip_force - 1000N plate_thickness 0mm plate_tolerance - 0mm
force1	Default forcedata when programming the SetForce instruction on the FlexPendant	tip_force - 1000N force_time 1s plate_thickness 0mm plate_tolerance - 0mm
reset_fault_time	Length of the reset pulse	0.5 s
reset_fault_idle	Wait time after reset pulse	0.5 s
flow_timeout{SW_NOF_GUNS}	Wait time for the water flow sensors to react	2 s
timer_ready_tout	Wait time for the timer ready signal	2 s
tipwear_superv	Soft equalizing specific data. Tip wear supervision value, max allowed tip wear. This data is used to supervise the tip wear and is used in MeasureWear and ReCalTcp instructions.	+- 0.2 mm

Name	Description	Default value
tipchg_superv	Soft equalizing specific data. Tip change supervision value, max allowed deviation of tip size. This data is used to supervise a missing tip or wrong size of the tip and is used in the MeasureWear and ReCalTcp instructions.	+- 3 mm
teach_dist	Soft equalizing specific data. Distance between programmed weld position and sheet surface	0 mm
opposite_z	Soft equalizing specific data. Defined z-direction for TCP, gives move direction for tip measurement	FALSE
m_search_speed	Soft equalizing specific data. Search speed during tip measurement (between 1 - 5 mm/s)	5 mm/s
m_touch_force	Soft equalizing specific data. Contact force (in N) during tip measurement (typically between 50 - 150N)	100 N
m_movein_dist	Soft equalizing specific data. Maximal distance from programmed point to search for reference surface	10 mm
mea_T1_filt	Soft equalizing specific data. Filter parameter for the tip measurement	0
mea_T2_filt	Soft equalizing specific data. Filter parameter for the tip measurement	0

5.3.4 Data definition routines

The following predefined routines are used to transfer user defined data to internally used data. They are used by the spotweld instructions and are called from the kernel during the process. Some of them are also called during system events such like poweron, program start, program stop etc.

These routines have a default functionality but can easily be changed. The routines cannot be deleted since they are called from internal modules.

PROC DefineSpotData(spotdata Spot, num GunNum)

This routine is called in the beginning of each Spot instruction.

Copy user spotdata to internal spot data, that is spotdata in the SpotL instruction.

PROC DefineGunData()

This routine is called just before execution of SwPowerOn, SwStart or SwReStart.

Copy user gun data to internal gun data, that is curr_gundata.

PROC DefineForceData(forcedata Force, num GunNum)

This routine is called in the beginning of each SetForce instruction.

Copy user forcedata to internal force data, that is forcedata in the SetForce instruction.

PROC DefineSimData()

This routine is called just before execution of SwPowerOn, SwStart or SwReStart.

Copy user simdata to internal simdata, that is curr_simdata.

PROC UpdateCalibData(num CurrTipWear, num RetPosAdj, num GunNum \switch ToolChg | switch TipChg | switch TipWear | switch FineCalib)

This routine updates the current tipwear in curr_gundata.

It is called at the end of each CalibL/J and Calibrate instruction.

PROC UpdateFixTipData(num CurrTipWear, num RetPosAdj,num GunNum, INOUT num status \switch Reference | switch TipChange | switch TipWear | switch RefChange)

This routine updates the current fixed tipwear in curr_gundata
It is called at the end of each MeasureWearL instruction.

This routine is only present if Software Equalizing.

PROC UpdateSpotData(z_int_spotdata Spot, num GunNum)

This routine updates the curr_spotdata.

It is called at the start of motion, that is when the robot starts to move.

Note!

Some of the routines only concerns servo guns and Software Equalizing.

5.4 SWSUP

5.4.1 Description

This module is intended for the person who does advanced customizing, and it contains the routines for the autonomous supervision and signal control.

SWSUP is running as a separate semistatic task, SW_SUP.

This chapter describes the default functionality.

For a multi spot system there will be one SW_SUP task configured for each robot in the system.

For example:

- SW_SUP1
- SW_SUP2

Note!

After changing any routines in SWSUP, the following steps must be taken before there is an affect on the application:

- Save SWSUP. The old one is overwritten.
- Generate a P-Start to affect all tasks

5.4.2 Contents

Data

The following interrupts and signal references are predefined:

After the routine DefSupIO and SupInit has been executed they will reference to the physical signal in the activated equipment and be connected to specific trap routines.

Name	Declaration
imotor_on{SW_NOF_GUNS}	LOCAL VAR intnum
imotor_off{SW_NOF_GUNS}	LOCAL VAR intnum
iwater_nokSW_NOF_GUNS}	LOCAL VAR intnum
iproc_ok{SW_NOF_GUNS}	LOCAL VAR intnum
iproc_error{SW_NOF_GUNS}	LOCAL VAR intnum
icurr_enable{SW_NOF_GUNS}	LOCAL VAR intnum
icurr_disable{SW_NOF_GUNS}	LOCAL VAR intnum
process_fault	VAR signaldo
enable_curr	VAR signaldo
start_water_	VAR signaldo
weld_power	VAR signaldo
process_run	VAR signaldo
water_ok	VAR signaldi

I/O Definition Routines and Supervision Init Routines

The following predefined routines are installed with the application.

These routines have a default functionality but can easily be changed.

PROC DefSupIO()

This routine is called at start (power on) from the main routine in this task.

Default functionality:

The signal names used in the event routines in this module are connected to the physical used signals. As default, signals for four gun equipment are defined.

PROC SupInit()

This routine is called at start (power on) from the main routine in this task.

Default functionality:
All signal interrupts are connected to trap routines here.

Note!
Some of the signals are connected System Output events, if these are changed the trap routines might not work like intended.

Event Routines

Default functionality with continuous water supervision:

The weld contactor is activated and the water is turned on from event routines activated by trap routines.

The following predefined routines are installed with the application.
These routines have a default functionality but can easily be changed.

Parameters description for the event routines:
GunNum: Gun equipment number

PROC SwMotorOn(num GunNum)

This routine is called when the system goes to Motors On state (motor_on = 1).

Default functionality:
If there is no process fault active and if no simulation is active then the weld contactor will be activated and the water will be started. If the water flow is not reached within a certain time (flow_timeout) the water will be turned off.

If these conditions are not true the weld contactor will be deactivated and the water will be turned off.

PROC SwMotorOff(num GunNum)

This routine is called when the system goes to Motors Off state (motor_on = 0).

Default functionality:
The weld contactor will be deactivated.

PROC SwProkOk(num GunNum)

This routine is called when the system goes into Process OK state (process_fault = 0).

Default functionality:

If the system is in motors on state and if no simulation is active then the weld contactor will be activated and the water will be started. If the water flow is not reached within a certain time (flow_timeout) the water will be turned off.

If these conditions are not true the weld contactor will be deactivated.

PROC SwProcError(num GunNum)

This routine is called when the system goes to Process Error state (process_fault = 1).

Default functionality:

The weld contactor will be deactivated and the water will be turned off.

PROC SwSupCurrEn(num GunNum)

This routine is called when the system goes into Weld state (enable_curr = 1).

Default functionality:

If the system is in motors on state and if no process fault is active simulation is active then the weld contactor will be activated and the water will be started. If the water flow is not reached within a certain time (flow_timeout) the water will be turned off.

If these conditions are not true the weld contactor will be deactivated and the water will be turned off.

PROC SwSupCurrDis(num GunNum)

This routine is called when the system goes into Simulation state (enable_curr = 0).

Default functionality:

The weld contactor will be deactivated.

6 I/O configuration

6.1 I/O configuration

Description

The digital and analog signals used for Spot Options are configured in the system parameters.

From the ABB menu on the FlexPendant:

- Tap **Control Panel**
- Tap **Configuration**
- Tap **Topics** and select **I/O**

The I/O configuration can also be accessed from RobotStudio.

The Spot package can be configured for different equipment setups. This chapter describes the different I/O setups available used by the Spot Options.

The physical connections can be changed freely. To save physical signals, signals not in use can be connected to a virtual unit (type Virtual).

6.2 Basic setup - simulated board description

Basic setup

The basic setup is for four gun equipment but it is possible to configure up to ten gun equipment in a similar way. This signal configuration is enough for running the default versions of the Spot packages.

Predefined I/O units

There are five predefined I/O units:

- One virtual unit, unit SW_BOARD1, with basic setup signals for gun equipment 1. Changing this unit to a physical digital I/O unit is enough for running the default version of the Spot Options package.
- One virtual unit, named SW_BOARD2 with basic setup signals for gun equipment 2. Change this unit to a physical digital I/O unit if two guns are used.
- One virtual unit, named SW_BOARD3 with basic setup signals for gun equipment 3. Change this unit to a physical digital I/O unit if three guns are used.
- One virtual unit, named SW_BOARD4 with basic setup signals for gun equipment 4. Change this unit to a physical digital I/O unit if four guns are used.
- One virtual unit, named SW_SIM_BOARD with some internal or normally not connected signals. Normally these signals will remain on a virtual unit.

6.3 Basic setup - signal description

Simulated weld timer signals for gun 1

Name	Type	Information
g1_start_weld	output	Start signal to the weld timer
g1_weld_prog	output group	Weld program number
g1_parity	output	Weld program number parity bit, (Placed on a virtual board).
g1_weld_power	output	The signal is set depending on the motor on state. Possible to use for a weld power unit contactor. See also System Module SWSUP in *Customizing* on page 237.
g1_reset_fault	output	Reset signal. Can be used to reset the welding controller after a weld error. The signal is pulsed with a user defined pulse length before manual or automatic rewelding.
g1_enable_curr	output	Signal used for the weld simulation function (simtype = 2). See *Programming* on page 15.
g1_weld_complete	input	Ready signal from the weld timer.
g1_weld_fault	input	Fault signal from the weld timer. If this signal is activated during the weld process the weld error handling is started without waiting for weld time out.
g1_timer_ready	input	The timer is ready to weld.
g1_new_program	output	Signal used to inform the weld timer that a new weld program has been sent. Some weld timers require this signal, for example Bosch. If not used this signal can be moved to a virtual board.

Simulated gun signals for gun 1

Name	Type	Information
g1_equalize	output	Gun equalize signal.
g1_close_gun	output	Gun close signal for a the pneumatic gun.
g1_open_hlift	output	Signal used to open the pneumatic gun to the highlift position.
g1_close_hlift	output	Signal used to close the pneumatic gun from the highlift position.
g1_gun_open	input	Signal indicating that the pneumatic gun is opened.
g1_hlift_open	input	Signal indicating that the pneumatic gun has reached the highlift position.
g1_pressure_ok	input	Signal indicating that the right gun pressure is reached for a pneumatic gun.
g1_start_water	output	Signal used to activate the water cooling system.
g1_temp_ok	input	Signal indicating over-temperature.
g1_flow1_ok	input	Signal indicating problems with the water supply in pipe 1.
g1_flow2_ok	input	Signal indicating problems with the water supply in pipe 2.
g1_air_ok	input	Signal indicating low air pressure in the equalize cylinder.
g1_weld_ contact	input	Signal indicating the state of the weld contactor (0 = deactivated).
g1_equipment_ok	input	Signal indicating the total gun status. A number of input signals from the gun is cross connected to this signal.
g1_press_group	output group	Pressure output for a pneumatic gun.

Simulated process status signals for gun 1

Name	Type	Information
g1_process_run	output	Is set at motion start and is reset when the weld process is ready and motion is released.
g1_process_fault	output	Is set when an error situation occurs and the process is interrupted.

Note!
Signal names for gun 2 are the same as for gun 1 but prefixed with g2_.

Other simulated signals

Name	Type	Information
force_complete	input	Can be used to interrupt the `SetForce` instruction before the programmed force time elapsed for a servo gun.
reweld_proc	input	Can be used to answer a weld error dialog with an input signal. The same as tapping **REWELD**.
skip_proc	input	Can be used to answer an error dialog with an input signal. The same as tapping **SKIP**.

6.4 SpotPack Bosch Compact AC weld timer

Default setup

The default setup is for two gun equipment, but it is possible to configure up to 10 gun equipment if needed.

Predefined I/O units

There are two predefined I/O units:

- One DeviceNet unit, named SW_BOARD1, with signals for the weld timer, media panel and the two gun equipment.
  This unit is configured on address 22 by default.
- One virtual unit, named SW_SIM_BOARD with some internal or normally not connected signals. Normally these signals will remain on a virtual unit.

Weld timer signals

Name	Type	Information
g1_weld_complete	input	Ready signal from the weld timer.
g1_weld_fault	input	Fault signal from the weld timer. If this signal is activated during the weld process the weld error handling is started without waiting for weld time out.
g1_timer_ready	input	The timer is ready to weld.
g1_tipdress_req	input	Fault signal from the weld timer. This signal is activated during the weld process when a tip dressing is required.
g1_tipchg_warn	input	Warning signal from the weld timer. This signal is activated during the weld process when a tips are almost worn out.
g1_tipchg_req	input	Fault signal from the weld timer. This signal is activated during the weld process when a tip change is required.
g1_start_weld	output	Start signal to the weld timer.
g1_enable_curr	output	Signal used for the weld simulation function (`sim-type = 2`). See *Programming* on page 15.
g1_reset_fault	output	Reset signal. Can be used to reset the welding controller after a weld error. The signal is pulsed with a user defined pulse length before manual or automatic rewelding.
g1_ack_tipdress	output	Reset signal for the tipdress request.
g1_ack_tipchg	output	Reset signal for the tipchange request.
g1_weld_prog	output group	Weld program number.

Gun signals

Name	Type	Information
g1_gun_open	input	Signal indicating that the pneumatic gun is opened.
g1_hlift_open	input	Signal indicating that the pneumatic gun has reached the highlift position.
g1_pressure_ok	input	Signal indicating that the right gun pressure is reached for a pneumatic gun.
g1_temp_ok	input	Signal indicating over-temperature.
g1_flow1_ok	input	Signal indicating problems with the water supply in pipe 1.
g1_flow2_ok	input	Signal indicating problems with the water supply in pipe 2. (This signal is optional)
g1_air_ok	input	Signal indicating low air pressure in the equalize cylinder. (This signal is optional)
g1_weld_ contact	input	Signal indicating the state of the weld contactor. (0 = deactivated) (This signal is optional)
g1_equipment_ok	input	Signal indicating the total gun status. A number of input signals from the gun is cross connected to this signal.
g1_equalize	output	Gun equalize signal.
g1_close_gun	output	Gun close signal for a the pneumatic gun.
g1_open_hlift	output	Signal used to open the pneumatic gun to the highlift position.
g1_close_hlift	output	Signal used to close the pneumatic gun from the highlift position.
g1_start_water	output	Signal used to activate the water cooling system.
g1_press_group	output group	Pressure output for a pneumatic gun.

Process status signals

Name	Type	Information
g1_process_run	output	Is set at motion start and is reset when the weld process is ready and motion is released.
g1_process_fault	output	Is set when an error situation occurs and the process is interrupted.

Other signals

Name	Type	Information
force_complete	input	Can be used to interrupt the SetForce instruction before the programmed force time elapsed for a servo gun.
reweld_proc	input	Can be used to answer a weld error dialog with an input signal. The same as tapping **REWELD**.
skip_proc	input	Can be used to answer an error dialog with an input signal. The same as tapping **SKIP**.

Note!

The signal names for gun 2 are the same as for gun 1 and they are mapped on the same physical I/O signals.

6.5 SpotPack Bosch Standard AC/MFDC weld timers

Default setup

The default setup is for four gun equipment but it is possible to configure up to ten gun equipment if needed.

Predefined I/O units

There are six predefined I/O units:

- One DeviceNet unit, named SWTimer1_PC1, with signals for the weld timer. This unit is configured on address 21 by default.
- One DeviceNet unit, named SW_BOARD1, with signals for gun equipment 1, media panel etc. This unit is configured on address 10 by default.
- One virtual unit, named SW_BOARD2 with signals for gun equipment 2. Change this unit to a physical digital I/O unit if two guns are used.
- One virtual unit, named SW_BOARD3 with signals for gun equipment 3. Change this unit to a physical digital I/O unit if three guns are used.
- One virtual unit, named SW_BOARD4 with signals for gun equipment 4. Change this unit to a physical digital I/O unit if four guns are used.
- One virtual unit, named SW_SIM_BOARD with some internal or normally not connected signals. Normally these signals will remain on a virtual unit.

Weld timer signals

Name	Type	Information
g1_weld_complete	input	Ready signal from the weld timer.
g1_weld_fault	input	Fault signal from the weld timer. If this signal is activated during the weld process the weld error handling is started without waiting for weld time out.
g1_timer_ready	input	The weld timer is ready to weld.
g1_weld_ contact	input	Signal indicating the state of the weld contactor. (0 = deactivated) (This signal is optional)
g1_start_weld	output	Start signal to the weld timer.
g1_enable_curr	output	Signal used for the weld simulation function in the weld timer (`simtype = 2`). See *Programming* on page 15.
g1_reset_fault	output	Reset signal. Is used to reset the welding controller after a weld error. The signal is pulsed with a user defined pulse length before manual or automatic rewelding.
g1_weld_power	output	The signal is set depending on the motor on state. This signal is controlling the weld power unit contactor if present. See also System Module SWSUP in *Customizing* on page 237.
g1_new_program	output	Signal used to inform the weld timer that a new weld program has been sent.
g1_weld_prog	output group	Weld program number.

Gun signals

Name	Type	Information
g1_gun_open	input	Signal indicating that the pneumatic gun is opened. (Spot Pneu)
g1_hlift_open	input	Signal indicating that the pneumatic gun has reached the highlift position. (Spot Pneu)
g1_pressure_ok	input	Signal indicating that the right gun pressure is reached for a pneumatic gun. (Spot Pneu)
g1_temp_ok	input	Signal indicating over-temperature.
g1_flow1_ok	input	Signal indicating problems with the water supply in pipe 1.
g1_flow2_ok	input	Signal indicating problems with the water supply in pipe 2. (This signal is optional)
g1_air_ok	input	Signal indicating low air pressure in the equalize cylinder. (This signal is optional)
g1_equipment_ok	input	Signal indicating the total gun status. A number of input signals from the gun is cross connected to this signal.
g1_equalize	output	Gun equalize signal.
g1_close_gun	output	Gun close signal for a the pneumatic gun. (Spot Pneu)
g1_open_hlift	output	Signal used to open the pneumatic gun to the highlift position. (Spot Pneu)
g1_close_hlift	output	Signal used to close the pneumatic gun from the highlift position. (Spot Pneu)
g1_start_water	output	Signal used to activate the water cooling system.
g1_press_group	output group	Pressure output for a pneumatic gun. (Spot Pneu)

Process status signals

Name	Type	Information
g1_process_run	output	Is set at motion start and is reset when the weld process is ready and motion is released.
g1_process_fault	output	Is set when an error situation occurs and the process is interrupted.

Note!
Signal names for gun 2 are the same as for gun 1 but prefixed with g2_

Other signals

Name	Type	Information
force_complete	input	Can be used to interrupt the SetForce instruction before the programmed force time elapsed for a servo gun.
reweld_proc	input	Can be used to answer a weld error dialog with an input signal. The same as tapping **REWELD**.
skip_proc	input	Can be used to answer an error dialog with an input signal. The same as tapping **SKIP**.

7 Motion control

7.1 Servo gun introduction

Additional axes

The robot controller has functionality to control additional axes configured as servo guns (other types of supported additional axes are track motion, positioners, conveyors etc.). All servo guns are handled as separate mechanical units. This means that before a servo gun may be moved, the mechanical unit to which it belongs must be activated. Several servo guns may be active at the same time.

Hardware overview

Servo gun axes are controlled by the drive module. Internal drive units are mounted inside a standard drive module (for example for a IRB 6600 with one servo gun or for a IRB 6600 with two stationary servo guns).

Motion servo gun parameters

A motion servo gun parameter file is installed in the controller for each servo gun. The parameter files are optimized designed concerning system behavior and motion/process performance.

It is possible to read and change most of the parameters from the Robot Studio application after installation, as well as from the FlexPendant.

With Spot Servo some gun specific system parameters may be updated temporarily directly in the robot program using the instruction STTune. This function will make tuning of gun parameters easier.

7.2 General motion control for servo guns

Introduction

The motion functionality described in this section is common for servo guns and most other types of additional axes. The description is however adapted for servo guns.

Activation/Deactivation

A servo gun may be activated when the robot and all additional axes have come to a standstill by using the RAPID ActUnit instruction. This means that the servo gun is controlled and monitored by the robot controller.

A servo gun is normally automatically activated directly after loading its parameters and starting up the system (activate at startup). It may be deactivated during program execution later.

If several guns are sharing one tool changer there will be no automatic activation at startup. When the connected gun is activated, it will not be possible to activate another gun until the first one is deactivated (mutual exclusion).

Deactivation of the gun is only needed if the gun has to be disconnected, for service or for a tool change. The deactivation will store the guns current position. This position will be restored when the gun is activated next time. Deactivation is performed with a DeactUnit instruction and this will also stop the control and monitoring of the axis.

Jogging

The position of the gun arm can be jogged with the joy stick (see *Operating manual - IRC5 with FlexPendant, Jogging additional axes*). The distance between the two tips is displayed in the jogging window, expressed in mm. An out of range supervision will stop the movement if the gun is reaching max stroke or min stroke. Min stroke is normally zero or a small negative value (gun tips closed to contact with each other).

Synchronous movements of robot and servo gun

Normally, as for other additional axes a servo gun axis is moved synchronous with the robot movements in such a way that both movements will be completed exactly at the same time. However, it can also be moved independent of the robot movements, for example when closing the gun tips with a force. But during normal movements (for example `MoveL`, `MoveJ`, `MoveC`) in program execution, the tool axis movement will be synchronized. The combined

path of robot and servo gun(s) will be repeatable and independent of programmed speed. The robot TCP path, will be the same irrespective of the programmed movements of the servo gun's movable arm.

A robtarget includes position data for additional axes which also will be set when a ModPos is performed.

Example:

- p10 is a robtarget RAPID data.
- p10.extax.eax_aThe position of the additional axis with *logical axis* 7.
- p10.extax.eax_bThe position of the additional axis with *logical axis* 8.
- p10.extax.eax_fThe position of the additional axis with *logical axis* 12.

Logical axis is a system parameter defined for each axis **(RobotStudio: Configuration Editor, Motion, Joint)**. The robot itself uses logical axes 1-6 and additional axes use 7-12. The user can change the logical axis number to fit the application. Only axes with unique logical axis numbers may be activated at the same time.

For a servo gun, the position is defined as the opening distance of the tips in mm.

The value 9E9 is defined for axes that are not activated.

Independent gun movement

The gun is in independent mode and can be moved to a specified independent position. During independent mode, the control of the servo gun is separated from the robot. The gun can be closed, opened, calibrated or moved to a new independent position, but it will not follow coordinated robot movements.

The instruction `IndGunMove` is used to set the gun in independent mode and thereafter move the gun to a specific independent position.

This mode can be reset by executing the instruction `IndGunMoveReset`.

Supervision during general motion control

An out of range supervision will stop the movement if the gun is reaching max stroke or if it is closed to contact with the tips (reaching min stroke).

Motion collision detection may be activated for the robot. There is also a separate motion supervision for each controlled axis, including the gun axis. This axis supervision will detect if the gun arm collides or get stuck. A motion error will occur and the motion will be stopped.

7.3 Asynchronous movements with force control

Introduction

The motion functionality described in this section is only valid for servo gun axes.

Opening and closing in general

The gun may be closed asynchronously (independent of current robot movement) to a predefined plate thickness and tip force. The closing will immediately start to run the gun arm to the expected contact position (thickness). The closing movement will interrupt an on-going synchronous movement of the gun. When the tips reaches the programmed plate thickness, the movement is stopped and there is an immediate switch from position control mode to force control mode. In the force control mode a motor torque will be applied to achieve the desired tip force.

The force remains constant until an opening is ordered.

Opening of the gun will reduce the tip force to zero and move the gun arm back to the pre-close position.

The gun opening may also take place while the robot is moving. But it is not possible if the robot movement includes a synchronized movement of the servo gun axis. A motion error, *Tool opening could not be synchronized with robot movement*, will occur.

Welding

A gun closing is done when performing a weld. The applied force may be taken from the weld timer or from a RAPID data (`spotdata`). During force build up, the thickness of the plates will be measured. The welding is started when the force is reached but only if the measured plate thickness is approved. When the weld is ready, the gun is immediately opened to the pre-close to position.

In the Spot Servo options, the closing, opening, thickness measurement, weld start and opening is integrated in the `Spot(M)L/Spot(M)J` instructions.

Squeezing without welding

A gun closing is also typically done when doing tip dressing or when changing tips. The force will be held constant for a certain time, and then the gun is opened up again.

In the Spot Servo options, the `SetForce` instruction will squeeze the gun with a specified force, thickness and during a specified time. `SetForce` takes a `forcedata` as argument where these values are defined. A thickness test is integrated in the instruction.

Supervision during asynchronous movements with force control

During the position control phase of the closing/opening, motion supervision is active for the servo gun to detect if the arm collides or gets stuck.

There is a maximum motor torque defined in the motion parameters for the gun that never will be exceeded in order to protect the gun from damage. If the force is programmed out of range according to the guns force-torque table, the output force will be limited to this maximum allowed motor torque and a motion warning will be logged.

During the force control phase, the motion supervision will supervise the gun position not to exceed a certain distance from the expected contact position. This distance, Forced on Position Limit, is defined in the motion gun parameters **(RobotStudio: Configuration Editor, Motion, Supervision)** and will typically depend on the flexibility of the gun arm. This supervision will protect the gun if for instance one tip is lost.

During the force control phase their is an active speed limitation which will limit the speed of the gun. The speed limit value is defined in the gun parameters (see the tuning chapter in *Application manual - Additional axes and stand alone controller*). The speed will be actively limited to increase further when the speed limit is reached. The speed limitation will give a controlled behavior of the gun when it is ordered to close to a position where the tips not are in contact, avoiding a hard impact when tip contact is established.

7.4 Tip management

Introduction

The tip management functionality will find and calibrate the contact position of the gun tips automatically. It will also update and monitor the total tip wear of the gun tips. The total tip wear for each gun is stored in a RAPID data (see *gundata - Equipment specific weld data* on page 138).

The tips are calibrated with special RAPID instructions. Typically, two gun closings will be performed during a calibration. The calibration may be done when the robot is standing still, see *Calibrate - Calibrate the gun* on page 113, or during a robot movement see *CalibL/CalibJ - Calibrate the gun during movement'* on page 108.

Three different types of calibrations are supported: tip wear, tip change and tool change. All three will calibrate the contact position of the tips. The total tip wear will however be updated differently by these methods:

Note!

If software equalizing is used there are other methods available for the tip wear compensation. See *Software Equalizing* on page 41.

Tip wear calibration

To be used after a tip dressing. The gun contact position is calibrated and the total tip wear of the gun is updated. The calibration movements are fast and the switch to force control mode will take place at the zero position.

Note: this method must only be used to make small positional adjustments (< 3 mm) caused by tip wear / tip dressing.

Tip change calibration

To be used after mounting a new pair of tips. The gun contact position is calibrated and the total tip wear of the gun is reset. The first calibration movement is slow in order to find the unknown tip collision position and switch to force control. The second calibration movement is fast. This calibration method will handle big positional adjustments of the gun.

This calibration may be followed by a gun closing in order to squeeze the tips in place (using the SetForce instruction). A new tip change calibration is then done to update possible positional differences after the tip squeeze.

Tool change calibration

To be used after reconnecting and activating a servo gun. The gun contact position is calibrated and the total tip wear of the gun remains unchanged. The first calibration movement is slow in order to find the unknown tip collision position and switch to force control. The second calibration movement is fast. This calibration method will handle big positional adjustments of the gun.

The method should always be used after reconnecting a gun since the activation will restore the latest known position of the gun, and that position may be different from the actual gun arm position; the gun arm may have been moved when disconnected. This calibration method will handle big positional adjustments of the gun.

Tip change requirement

The total tip wear of the gun (stored in RAPID gundata) may be supervised in order to detect when a tip change is needed.

Tool center point adjustment

Half the total tip wear may be used to adjust / optimize the tool center point of the robot tool (RAPID tooldata).

Supervision during tip calibration

The same supervision will be active during calibration as during asynchronous movements with force control.

7.5 Installation and Service

Install servo gun parameters

If the system is cold booted, the servo gun may not be installed. Load the gun parameters from FlexPendant via Control Panel/Configuration/File using ‘Add new parameters’ and restart the system. Activate the gun in order to control and monitor the axis (note: the gun may be setup activated at startup).

The servo gun parameters may also be loaded from RobotStudio Controller/Configuration/ Load Parameters.

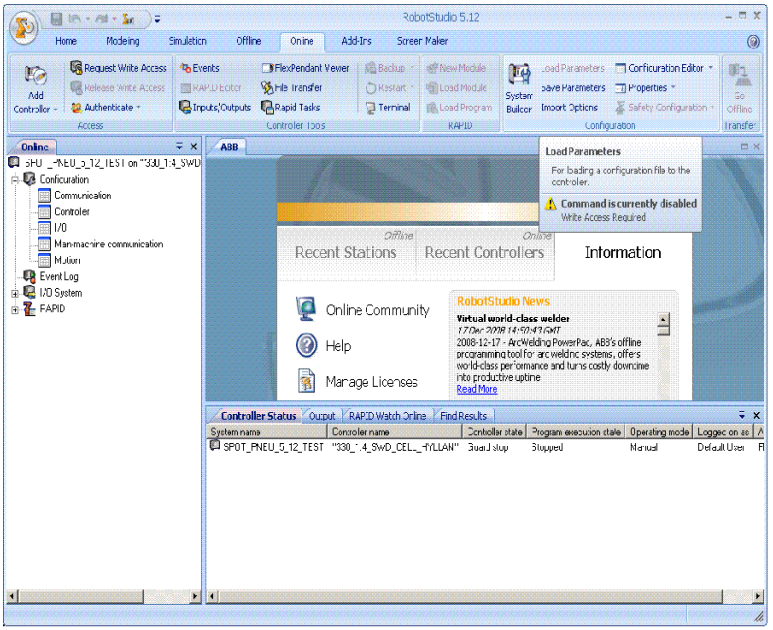


Figure 30 Install servo gun parameters

Set the servo gun name

After the gun parameters has been installed and the system has been restarted, the gundata needs to be updated with the servo gun name (mechanical unit name).

	Action	Note
1.	Run the service routine **ManAddGunName** to find all configured servo guns in the system, and add their names to the gundata array (curr_gundata). See *gundata - Equipment specific weld data* on page 138	Follow the instructions in the routine.
2.	Ready.	

Service calibration

After installing the gun parameters and restarting the system, the gun must be synchronized by performing a fine calibration. Apart from other kinds of additional axes, this action also requires running a RAPID service routine, **ManServiceCalib** to initialize or synchronize the gun position.

Note!

Check that a force calibration has been done for the gun before running this service routine. A force calibration should be done to calibrate the motor torque vs tip force characteristics. This calibration needs to be done in order to protect the servo gun from too high forces. It will not be possible to run any Spot instructions until a service calibration has been done. If the gun is **not** force calibrated and properly tuned, follow the tuning procedure described in *Application manual - Servo gun tuning.*

In the Program Editor, tap **Debug** and then tap **Call Service Routine**.

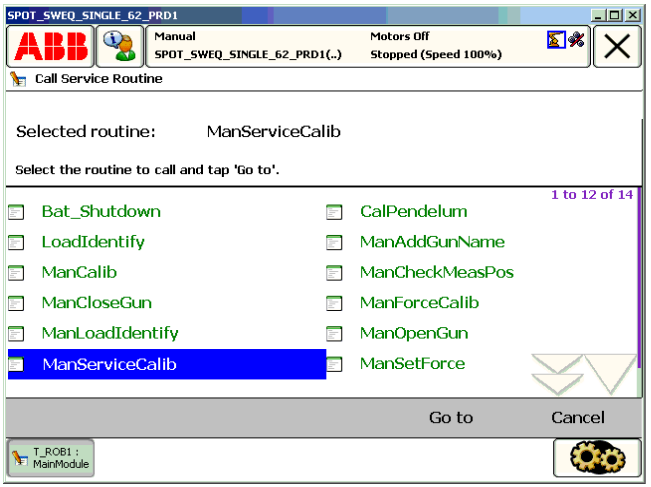


Figure 31 Service calibration

Fine calibration

Fine calibration must be performed when installing a new servo gun or if the servo gun axis is in state *Not Calibrated*. These steps are required:

	Action	Note
1.	Fine calibrate the axis in the Calibration window from the menu ABB -> Calibration (there is no need to jog the axis to any particular position)	ABB/Calibration/Mechanical Unit, Fine Calibration

	Action	Note
2.	Run the service routine **ManServiceCalib** option 2. If the gun is considered to be force calibrated (force_calib = true) the gun will move to zero position and close slowly until tip contact is detected. Otherwise a warning dialog will be displayed with a question whether the gun has been force calibrated or not. Answering **Yes** will perform the service calibration anyway, **No** will end the routine. **NOTE!** If the gun is **not** force calibrated and properly tuned, follow the tuning procedure described in *Application manual - Servo gun tuning.*	Follow the instructions in the routine.
3.	As a result, the gun position is updated to be zero in the position of contact and the tip wear value is reset.	
4.	Ready.	

Warning!

If the boolean **force_calib** is set to **TRUE** without having performed the force calibration procedure, the servogun can be damaged, please make sure that the servo gun is force calibrated and tuned and that the force calibration values are correct before setting this parameter, see *Application manual - Servo gun tuning.*

Update the revolution counter

An update of the revolution counter must be performed if the position of the axis is lost. If this happens, this is indicated by the calibration state *Rev. Counter not updated*. These steps are required to update the counter.

	Action	Note
1.	Update the revolution counter in the Calibration window from the menu ABB -> Calibration (there is no need to jog the axis to any particular position).	ABB/Calibration/Mechanical Unit, Update Revolution Counters

	Action	Note
2.	Run the service routine **ManServiceCalib** option 1.	If the gun is considered to be force calibrated (force_calib = true) the gun will move to zero position and close slowly until tip contact is detected, since the zero position is unknown. Otherwise a warning dialog will be displayed with a question whether the gun has been force calibrated or not. Answering **Yes** will perform the service calibration anyway, **No** will end the routine. **NOTE!** If the gun is **not** force calibrated and properly tuned, follow the tuning procedure described in *Application manual - Servo gun tuning.*
3.	As a result, the gun position is updated an integer number of revolutions to be zero in the position of contact. Tip wear of the gun remains unchanged.	
4.	Ready.	

Warning!

If the boolean **force_calib** is set to **TRUE** without having performed the force calibration procedure, the servogun can be damaged, please make sure that the servo gun is force calibrated and tuned and that the force calibration values are correct before setting this parameter, see *Application manual - Servo gun tuning*.

Force calibration

There is a RAPID service routine to calibrate the motor torque vs tip force characteristics, ManForceCalib. A separate sensor is needed to measure the tip force. An optional number of force recordings (2-10) is made where measured tip force is inserted with corresponding motor torque.

This calibration is done to protect the gun from to high forces.Tap **Debug** and then tap **Call Service Routine**

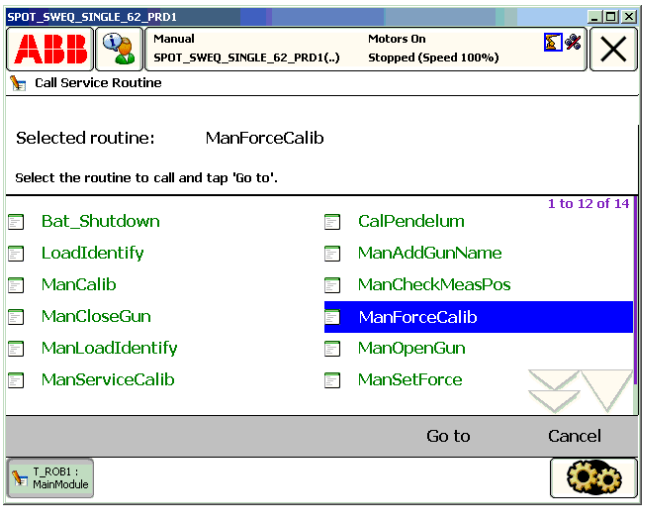


Figure 32 ManForceCalib

	Action	Note
1.	Run the service routine **ManForceCalib,** select option 1.	Change the force calibration setup data, for example the number of calibrations and the max force to be used during the calibration.
2.	Run the service routine **ManForceCalib,** select option 2. This will perform the calibrations	Follow the instructions in the routine.
3.	Ready.	

Disconnect/Reconnect a servo gun

If the servo gun is deactivated, using the `DeactUnit` instruction, it may be disconnected and removed. The gun position at deactivation will be restored when the gun is connected and reactivated. Make a tool change calibration to make sure the tip position is OK.

Recover from accidental disconnection

If the motor cables are disconnected by accident when the servo gun is activated, the servo gun must be deactivated in order to jog the robot to a service position. To deactivate, tap **Deactivate** in the Jogging window. After service or repair the revolution counter must be updated since the position has been lost.

NOTE!

To be able to continue program execution after repair, the service routine *ManServiceCalib* must be run. Option 1 if the revolution counters have been updated or option 2 if the gun is fine calibrated after the re-connection.

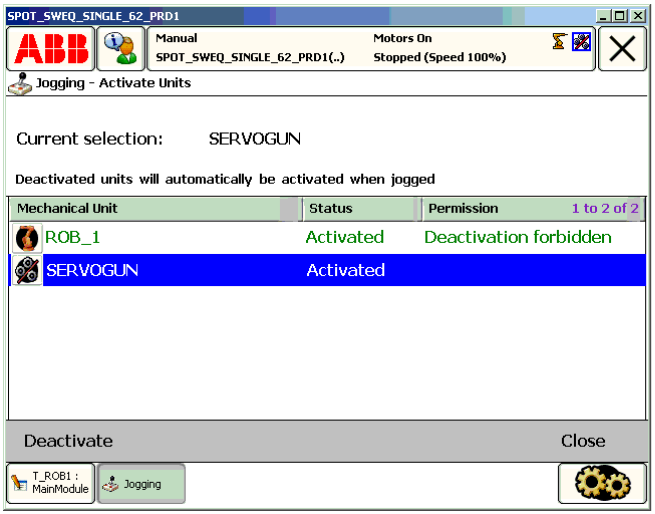


Figure 33 Accidental disconnection

Replace a servo gun

Install the parameters for the spare gun. The spare gun must have same parameter names as the original gun, otherwise the installation will just add the new gun, keeping the old gun in parallel. If the spare gun has identical parameters, it may run directly without installing new parameters and performing a fine calibration.

7.6 Stationary gun

A stationary servo gun is mounted on the floor and the robot is holding the work piece. The only difference when using a stationary servo gun is that the robot tool (RAPID `tooldata`) should be defined as stationary, and the used work objects as robot held.

7.7 Servo tool change

It is possible to change servo gun during production. The functionality is realized as the option *Servo Tool Change*. There is no software limitation in how to combine different kinds of servo guns (for example brands, sizes or motors) with a tool changer.

The used servo guns share the same drive unit, and the same node on the measurement board. They are activated as different mechanical units, but of course never at the same time. They may use the same or different logical axis.

Changing gun requires a deactivation of the operating gun and then unplugging it's motor cables. The motor cables are plugged in to the next gun, and this gun is activated and ready to run. The plug-in mechanism requires a mechanical tool changer interface to the guns. One individual set of gun parameters are installed for each gun.

Up to 8 additional axes (servo guns or other axes) can be installed simultaneously in one robot controller. All or some of them may be servo guns sharing a tool changer.

NOTE!

To be sure that the right servogun is activated and have a safe way to tool change, it is recommended that the connection relay functionality is used when tool changing.

The *Motion* parameters should be set like this when tool changing.

1. *Mechanical Unit/MecUnitName/Activate at StartUp* should be set to No.
2. *Mechanical Unit/MecUnitName/Deactivate Ptc at Disconnect* should be set to Yes.
3. *Mechanical Unit/MecUnitName/Use Connection Relay* should have the "MecUnitName".
4. *Relay/UnitName/Input Signal* should be a signal defined in EIO.cfg,

For example *diMecUnitName*, and this input signal should be connected to a sensor on the toolstand or the toolchanger.

5. *Measurement Channel/MecUnitName/Disconnect at Deactivate* should be set to Yes.

If this setup is used a safe toolchange functionality will be achieved.

NOTE!

Tool changing with servo guns requires the option *Servo Tool Change*.

Example, tool change procedure

The procedure to switch between gun 7A and gun 7B must includes these minimal actions (excluded here is the needed communication with the tool changer, the tool stand and necessary robot movements):

1	Deactivate gun SGUN1.	(The position of gun SGUN1 is stored)
2	Disconnect gun SGUN1.	(Disconnect the servo gun motor cables)
3	Connect gun SGUN2.	(Connect the motor cables to motor SGUN2)
4	Activate gun SGUN2.	(The latest position of gun SGUN2 is restored)
5	Run a tool change tip calibration of gun SGUN2.	(Make sure the position is correct)

WARNING!

If the servo gun axis has been moved during deactivation, the position of the axis might be wrong after activation, and this will not be detected by the controller. The position after activation will be correct if the axis not has been moved, or if the movement is less than 0.5 motor revolutions. Always use the tool change tip calibration after activation. The tool change calibration will adjust any positional error caused by gun movements during deactivation.

WARNING!

It is important that no other mechanical units used with one tool changer, are activated but the one corresponding to the currently connected servo gun! An activation of wrong mechanical unit may cause unexpected movements or errors. Some tool changers support IO signals that specifies which gun is currently connected. That information may be used to make sure correct mechanical unit is activated. It is also possible to lock activation of not connected mechanical units by specifying a digital input (DI) in the connection relay which is a motion system parameter **(RobotStudio: Configuration, Motion, Relay)** for each servo gun. This digital input, which also is setup in the EIO.cfg, is read when the mechanical unit is activated. If set to 1 the activation will take place normally, otherwise a recoverable motion error will occur and the activation will be denied.

7.8 Other references

Application manual - Spot Options	CalibL, CalibJ, Calibrate, SpotL, SpotJ, SetForce, STTune, gundata, spotdata, forcedata
Operating manual - IRC5 with FlexPendant	General motion control and programming
Technical reference manual - RAPID Instructions, functions and data types	ActUnit, DeactUnit, MoveL, MoveJ, robtarget, tooldata
Application manual - Servo gun tuning	How to tune a servogun.
Application manual - Additional axes and stand alone controller	Hardware: motors, resolvers, drives etc. Servo gun parameters, tuning a servogun.

8 FlexPendant

8.1 FlexPendant Interface

Introduction

This chapter describes the user interface intend to simplify the use of the spot welding functionality. The user have the most common data and signals collected together in one place, easy to use and understand. This is not a replacement for the standard FlexPendant functionality it is only a complement. To start RobotWare Spot, tap the **ABB** menu and then tap **RobotWare Spot**. The RobotWare Spot desktop is displayed, from here all spot welding functions can be accessed. For further information on using the FlexPendant the *Operating manual - IRC5 with FlexPendant* is strongly recommended.

Figure 34 RobotWare Spot desktop

RobotWare Spot has four function buttons:

- Status
- Process Signals
- Process Data
- Manual Actions

After the following short overview a more detailed description of each view will follow.

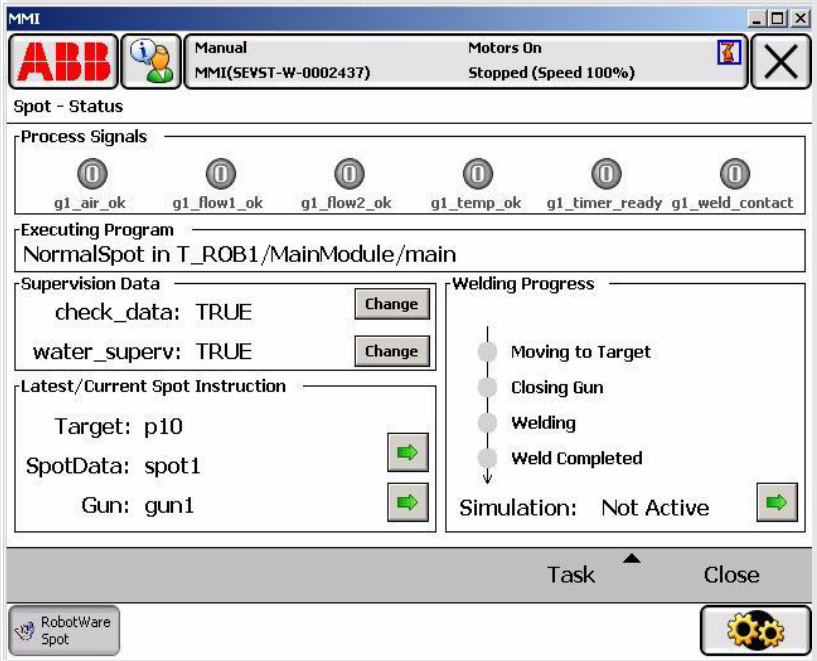
Status provides basic information about the currently executing spot program. You can also follow the welding process in every instruction by look in the welding process part of the window. It is also possible to quickly change some of the most common data used in a spot instruction by tapping the green arrow beside the data, this is a short cut to the process data window.

Process Signals displays the input and output signals controlling the spot process equipment. Digital signals are show in an illustrative way, reflecting the real equipment of the specific spot system in use. When testing the system in manual mode, it is possible not only to view but also to set new values to process signals.

Process Data presents spot specific data types and lists all instances of a selected data type, thus offering a quick and easy way to edit or view simdata, forcedata, gundata, or spotdata in order to improved the system performance.

Manual Actions provides all available manual service routines in the system. The user can easily start any routine by tapping the button for the routine he would like to run.

8.2 Status window

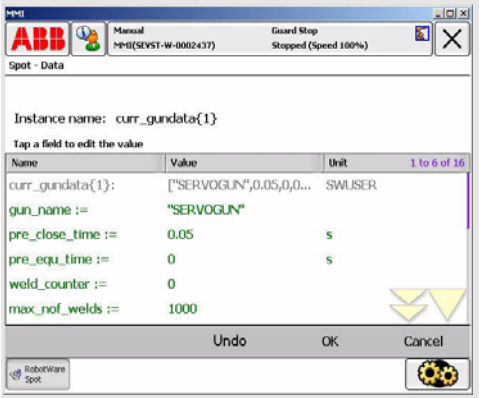
	Action
1.	Tap **Status** on the RobotWare Spot desktop. A window will appear containing functionality to follow the welding progress step by step, view the latest/current spot instruction, see the name and path to the executing weld program, disconnect and connect the supervision values for water and check data and also see the state of system signals connected to the welding equipment. This can be useful during programming, testing or production phase. 
2.	Tap **Task** to view another spot welding robot (if MultiMove system).
3.	Tap **Close** in the Status window to close the window and get back to the RobotWare Spot desktop.

How to change supervision data

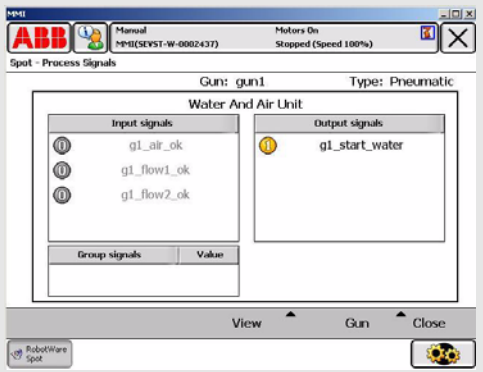
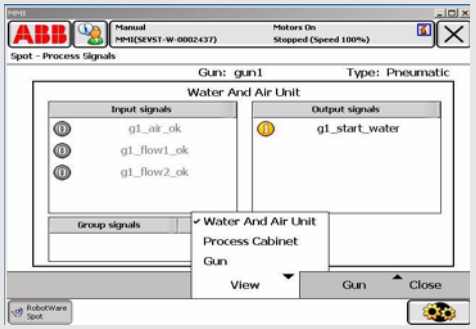
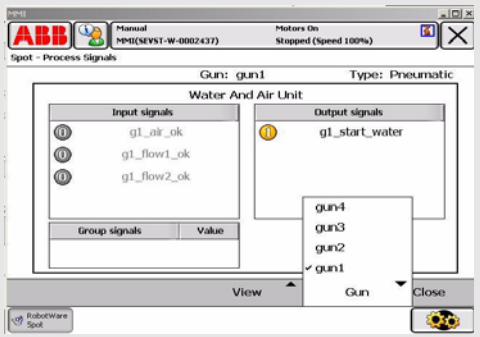
	Action	Note
1.	Select the data to be changed.	
2.	Tap **Change** icon beside the selected data.	
3.	In the appeared message box tap **OK** to close the message box and save the changed value.	**Cancel** closes the message box and no changes will be made

For more information about the supervision data see *RAPID references* on page 73, and *Customizing* on page 237.

How to view or edit data

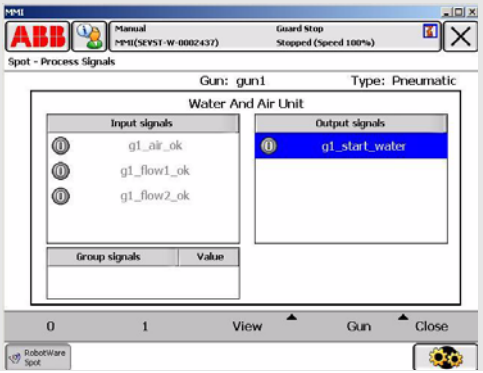
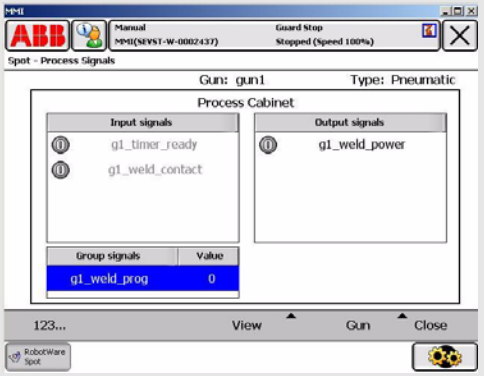
	Action
1.	Select the data to be changed.
2.	Tap the **Green arrow** icon beside the selected data
3.	View or edit value 
4.	Tap **Undo** to reset the window from all changes.
5.	Tap **Cancel** to return to the Status window without any changes.
6.	Tap **OK** to write the new value to the controller and to leave the Data dialog and return to the Status window.

8.3 Process Signals window

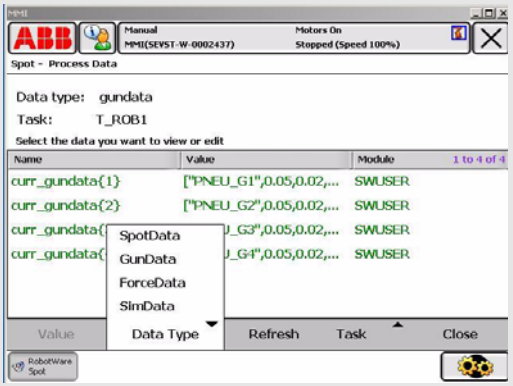
	Action	Note
1.	Tap the **Process Signals** icon in the RobotWare Spot desktop. A window will appear containing functionality for view digital input signals connected to different parts of the equipment and to set/reset some output and group signals	 Figure 35 Process Signals window
2.	Tap **View** to change to other parts of the equipment	 Figure 36 Process Signal window part selection
3.	Tap **Gun** to switch between all available guns in the system	 Figure 37 Process Signals window gun selection

	Action	Note
4.	Tap **Close** in the Process Signals window to close the window.	Get back to the RobotWare Spot desktop.

How to set/reset process signals

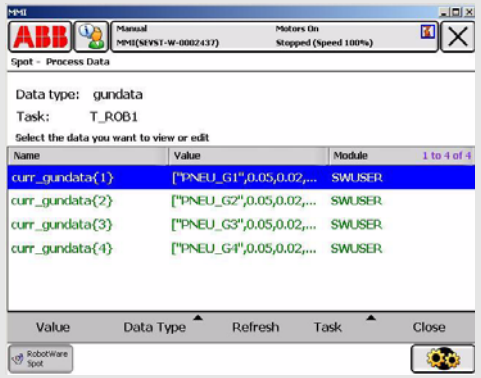
	Action	Note
1.	Select the signal to be changed	
2.	Depending if it is an output or a group signal to change you get different behavior in the command bar.	
3.	To change an output signal toggle between **1** and **0** to set/reset the signal.	
4.	To change a group signal tap **123...** in the command bar to open a numeric pad to written in a new value. When ready tap **OK** to close the pad.	

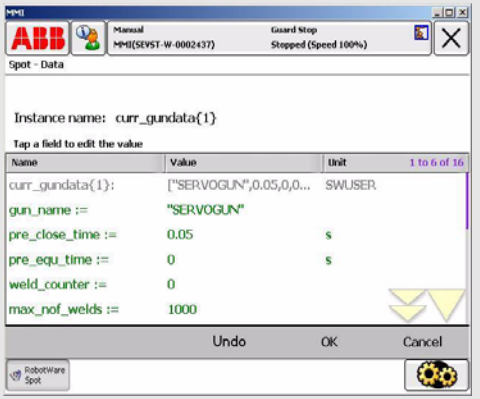
8.4 Process Data window

	Action	Note
1.	Tap the **Process Data** icon in the RobotWare Spot desktop A window will appear containing functionality for view and edit data types connected to the spot welding functionality. I it is a MultiMove system it is possible to view data from different tasks by tap Task in the command bar. To update the window and do a new search again for selected data type tap **Refresh** in the command bar.	 Figure 38 Process Data window

Data types

The data types available from the Data Types menu are simdata, forcedata, gundata and spotdata. Each type is thoroughly described in the *Programming* on page 15.

	Action	Note
1.	Tap **Data Type**. Select the data type you want to view or edit	
2.	To change the data of the listed data type, tap the data.	 Figure 39 Data window

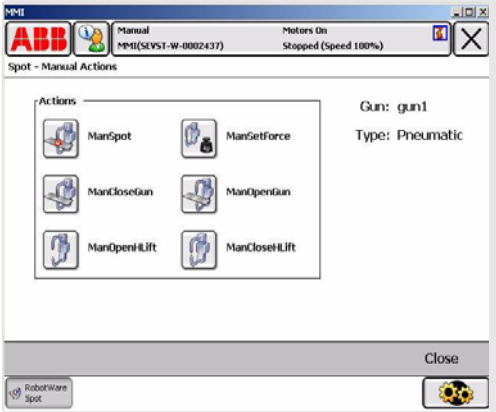
	Action	Note
3.	Tap **Value**. The declaration of the data is displayed.	 Figure 40 Data window

How to edit data

	Action
1.	Select the data instance you wish to edit from the list.
2.	Select the component you want to edit in the Data dialog.
3.	Depending on the component data type, enter the new value in the numeric or alphabetic pad that will appear.
4.	When ready tap **OK** button to close the pad.
5.	Tap **Refresh** to cancel all changes and start from the beginning again.
6.	Tap **OK** to write the new value to the controller and to leave the Data dialog.

More information on the individual components can be found in `spotdata`, `gundata`, `forcedata` and `simdata` descriptions.

8.5 Manual Actions window

	Action	Note
1.	Tap the **Manual Actions** icon in the RobotWare Spot desktop. A window will appear containing all available manual service routines in the system.	

How to start a service routine

	Action
1.	Set the system in MotorsOn state.
2.	Tap the service routine name you would like to run.

How to use the manual service routines is described in *Programming* on page 15.

8.6 Manual vs Automatic mode

If the system is in manual mode you can view or edit data and set or reset output signals. When you switch to automatic mode you loose the possibility to edit data and set or reset output signals but you can still view the current data or output signals.

8.7 Simulation

If you change the sim_type in simdata, when the value is separated from 0 a little stop sign icon is visible at the top right corner in every window. You can also see that the text change from **Not Active** to **Active** in the status window. See graphic below.

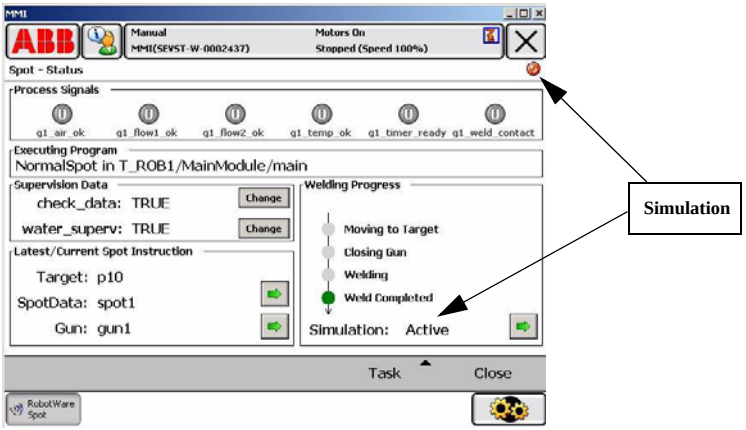


Figure 41 Simulation active

8.8 Configuration file

All systems includes a configuration file for the process signals shown in the application. To be able to add signals or change the default signal names, you must edit the file *SpotSystem.xml* (read the instruction inside). This file is copied to the Spot directory in the system together with the customized rapid modules.

See Customizing RobotWare-SpotServo *How to use own signal names on internal used signals* on page 248

Note!

If the I/O signal names listed in SpotSystem.xml does not match the actual I/O configuration the Spot MMI, FlexPendant Interface will not work correctly.

	Described in:
Customizing RobotWare-SpotServo	*Customizing* on page 237

9 Bosch FlexPendant

9.1 Bosch FlexPendant Interface

Introduction

This chapter describes the user interface intend to simplify the use of the Bosch weld timer functionality. To access the Bosch FlexPendant interface you need the software option *Bosch Weld Timer Interface*. You also need recommended hardware (weld timer) etc.

The operator has the most common data and weld timer errors and fault collected together in one place, easy to use and understand.

To start Bosch Timer tap the **ABB** menu and then tap **Bosch Weld Timer**. The Bosch Timer desktop shows all common weld timer functions. For further information on using the FlexPendant, see *Operating manual - IRC5 with FlexPendant*. For information about weld parameters, see *Bosch operating and programming manual volume 2*.

Figure 42 Bosch Timer desktop

Bosch Timer desktop has four function buttons:

- Pre Warning
- Weld Fault
- Last Weld
- Weld Parameters

After the following short overview a more detailed description of each view will follow.

Pre Warning present information about all electrodes configured for a certain weld timer.

Weld Fault lists all weld faults and warnings connected to the welding process and the weld timer.

Last Weld present information about the last weld performed by the weld timer.

Weld Parameters offering a quick and easy way to view or edit ordinary weld parameters.

General information before start using the Bosch interface

Note!

Before using the Bosch application some necessary setup is needed, use PC software BOS5000/BOS6000.

- Transformer parameter setup.
- Force and current calibration of the electrodes.

Note!

If or when an extra ordinary weld fault (hardware fault) occurs the user have to connect the BOS5000/BOS6000 pc-software to get the real cause of the problem.

Note!

To do a backup or restore the weld timer the user have to connect the BOS5000/BOS6000 pc-software to take this action.

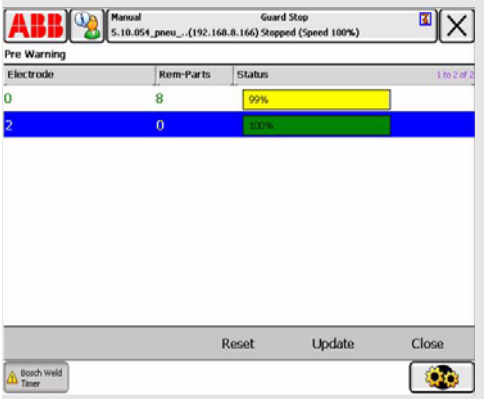
Note!

Make sure you understand what happens to the welding sequence if you turn off the Ignition parameter under the Settings or General node on the FlexPendant, or if you change the simulation type in RobotWare Spot.

Note!

Avoid to close the application during loading or saving weld parameters in the Weld Parameters window, this can lead to loss of data.

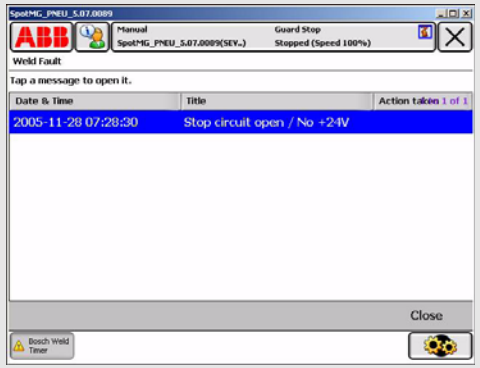
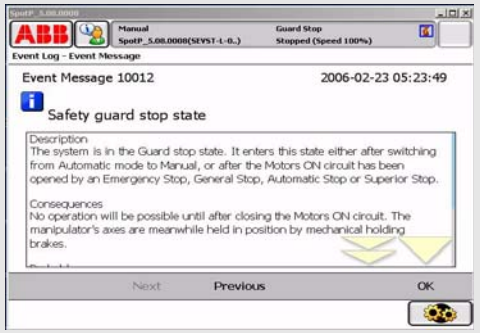
9.2 Pre Warning window

	Action	Note
1.	Tap **Pre Warning**. A window will appear containing information of all configured electrodes in the weld timer.	 Figure 43 Pre Warning window
2.	Tap **Reset** to reset the value of the selected electrode (not enable if no row selected in the list view).	
3.	Tap **Update** to force the view to update itself and search for new information about the electrodes.	
4.	Tap **Close** in the Pre Warning window to close the window and return to the Bosch Timer desktop.	

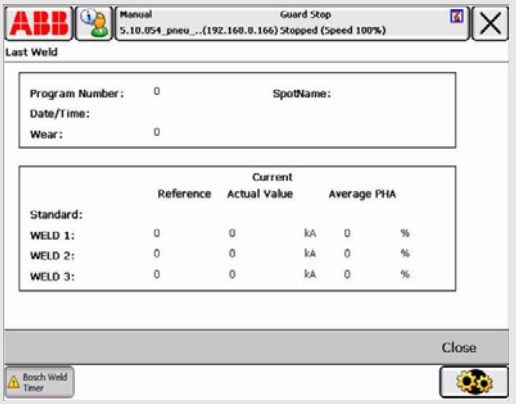
How to reset the value of an electrode

	Action	Note
1.	Select the electrode in the list view to be changed.	
2.	Tap **Reset** in the command bar.	

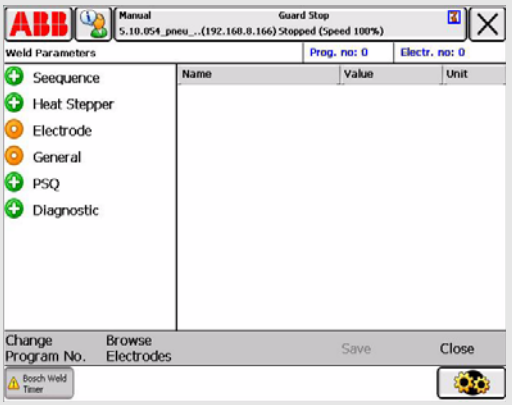
9.3 Weld Fault window

	Action	Note
1.	Tap **Weld Fault**. A window will appear containing information about all errors and warnings connected to the welding process and the weld timer.	 Figure 44 Weld Fault window
2.	Tap twice on a row in the list view to open up a new window for more information about the weld error or warning.	 Figure 45 Weld Fault window
3.	Tap **Close** in the Weld Fault window to close the window and return to the Bosch Timer desktop or **OK** to return to the previous window.	

9.4 Last Weld window

	Action	Note
1.	Tap **Last Weld**. A window will appear containing information about the last weld performed by the weld timer.	 Figure 46 Last Weld window
2.	Tap **Close** in the Last Weld window to close the window and return to the Bosch Timer desktop.	

9.5 Weld Parameters window

	Action	Note
1.	Tap **Weld Parameters**. A window will appear containing information about the weld parameters that is possible to view and edit in the weld timer (this view is not possible to use in automatic mode).	 Figure 47 Weld Parameters window
2.	Tap **Change Program No.** to view weld parameters for different programs in the weld timer.	
3.	Tap **Browse Electrodes** to change electrode number.	
4.	Tap **Close** to return to the Bosch Timer desktop.	
	Note! Before changing any parameters in this view you have to had good knowledge about the welding equipment and the welding parameters, otherwise it is easy to damage or destroy the welding equipment.	

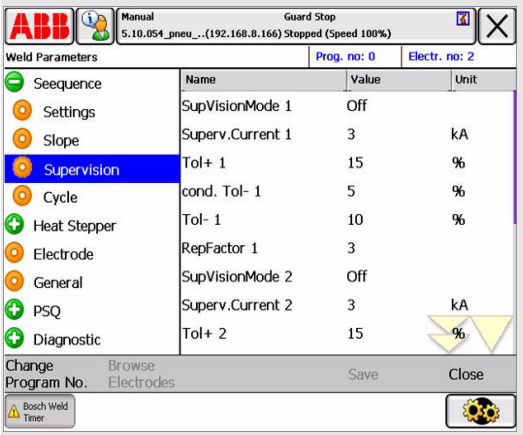
Cycle node

The cycle node contains parameters related to the welding sequence.

	Action	Note
1.	See Bosch weld timer manual volume 2 (page8-10) for more information about the parameters.	 Figure 48 Cycle node
2.	Tap **Change Program No.** to view weld parameters for different programs in the weld timer.	
3.	Tap **Save** to save the changes done to the parameters and stay in current window.	
4.	Tap **Close** to close the window and return to the previous window.	

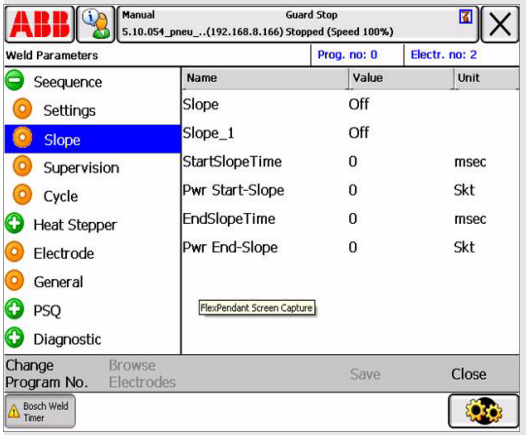
Supervision node

The supervision node contains parameters related to the supervision of the current.

	Action	Note
1.	Each weld time (1, 2, 3) can be supervised separately. See the Bosch weld timer manual volume 2 (page11-29 and 11-36) for more information about the parameters.	 Figure 49 Supervision node
2.	Tap **Change Program No.** to view weld parameters for different programs in the weld timer.	
3.	Tap **Save** to save the changes done to the parameters and stay in current window.	
4.	Tap **Close** to close the window and return to the previous window.	

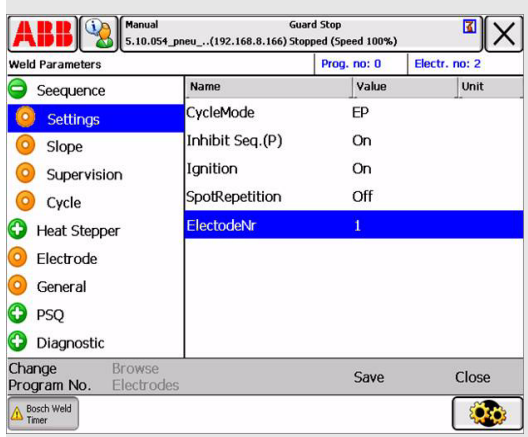
Slope node

The slope node contains parameters related to the welding sequence, if up and down slope of the current is required changes are done here.

	Action	Note
1.	See Bosch weld timer manual volume 2 for more information about the parameters.	 Figure 50 Slope node
2.	Tap **Change Program No.** to view weld parameters for different programs in the weld timer.	
3.	Tap **Save** to save the changes done to the parameters and stay in current window.	
4.	Tap **Close** to close the window and return to the previous window.	

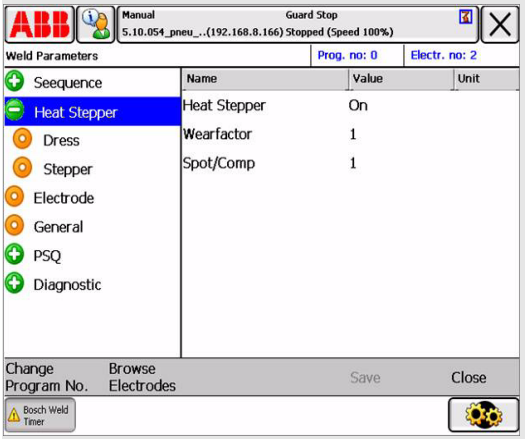
Settings node

The settings node contains parameters related to the selected welding program.

	Action	Note
1.	See Bosch weld timer manual volume 2 for more information about the parameters.	 Figure 51 Settings node
2.	Tap **Change Program No.** to view weld parameters for different programs in the weld timer.	
3.	Tap **Save** to save the changes done to the parameters and stay in current window.	
4.	Tap **Close** to close the window and return to the previous window.	

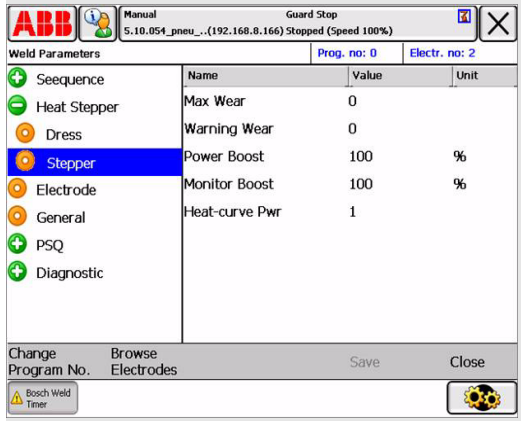
HeatStepper node

The heatstepper node contains parameters related to the wear of the electrodes, if the customer use this parameters changes are done continuously.

	Action	Note
1.	See Bosch weld timer manual volume 2 for more information about the parameters.	 Figure 52 HeatStepper node
2.	Tap **Change Program No.** to view weld parameters for different programs in the weld timer.	
3.	Tap **Browse Electrodes** to change electrode number.	
4.	Tap **Save** to save the changes done to the parameters and stay in current window.	
5.	Tap **Close** to close the window and return to the previous window.	

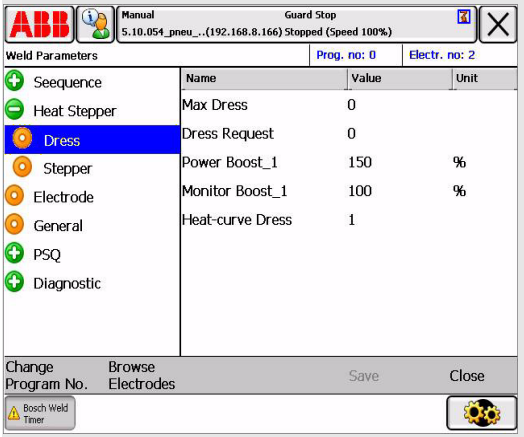
Stepper node

The stepper node contains parameters related to the wear of the electrodes, if the customer use this parameters changes are done continuously.

	Action	Note
1.	See Bosch weld timer manual volume 2 for more information about the parameters.	 Figure 53 Stepper node
2.	Tap **Change Program No.** to view weld parameters for different programs in the weld timer.	
3.	Tap **Browse Electrodes** to change electrode number.	
4.	Tap **Save** to save the changes done to the parameters and stay in current window.	
5.	Tap **Close** to close the window and return to the previous window.	

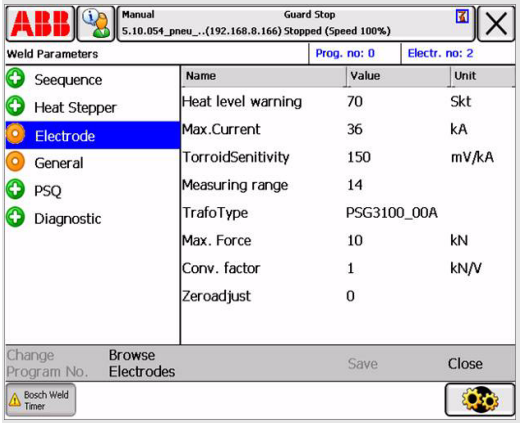
Dress node

The dress node contains parameters related to the wear of the electrodes, if the customer use this parameters changes are done continuously.

	Action	Note
1.	See Bosch weld timer manual volume 2 for more information about the parameters.	 Figure 54 Dress node
2.	Tap **Change Program No.** to view weld parameters for different programs in the weld timer.	
3.	Tap **Browse Electrodes** to change electrode number.	
4.	Tap **Save** to save the changes done to the parameters and stay in current window.	
5.	Tap **Close** to close the window and return to the previous window.	

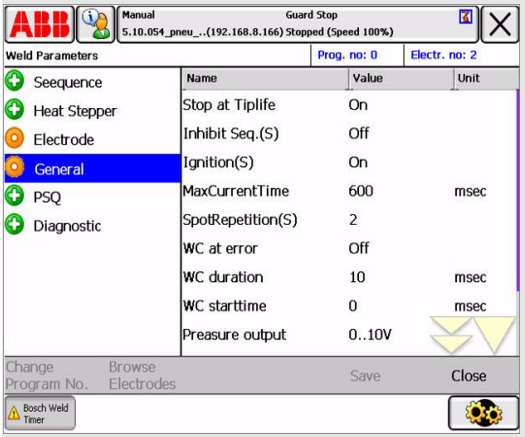
Electrode node

The electrode node contains parameters related to the electrode (electrode number), changes to this parameters are done in the start phase or when a new gun is initiated.

	Action	Note
1.	See Bosch weld timer manual volume 2 for more information about the parameters.	 Figure 55 Electrode node
2.	Tap **Browse Electrodes** to change electrode number.	
3.	Tap **Save** to save the changes done to the parameters and stay in current window.	
4.	Tap **Close** to close the window and return to the previous window.	

General node

The general node contains parameters related to the complete weld timer. They are normally set up in the start up phase.

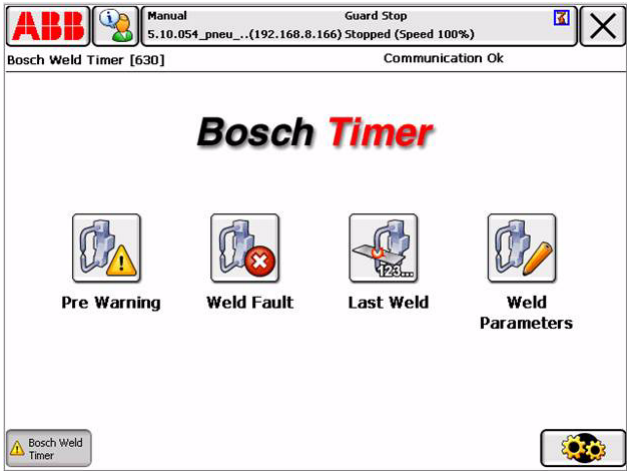
	Action	Note
1.	See Bosch weld timer manual volume 2 for more information about the parameters.	 Figure 56 General node
2.	Tap **Save** to save the changes done to the parameters and stay in current window.	
3.	Tap **Close** to close the window and return to the previous window.	

9.6 Manual vs Automatic mode

If the system is in manual mode you can view or edit weld parameters and see information about the warning and error that can occur in the system. When you switch to automatic mode or when executing a program you cannot open the Weld Parameters window but you can still open the other windows.

9.7 General information

The information at the upper right side shows if the communication with the timer is ok or not (if problem see the system Event Log for further information). In the upper left side you have the information about the used device net protocol for the connected timer.



9.8 SIO configuration for Bosch Weld Timer Interface (sio.cfg)

Currently it is only possible to connect one weld timer to the robot controller, this communication is done via an RS232 cable connected from the controller to the weld timer.

In the future it will be possible to connect several weld timers to the controller via Ethernet, this requires an optional Ethernet card in the weld timer.

Warning!

Avoid to disconnect the RS232 cable during loading or saving weld parameters in the Weld Parameters window, the Bosch MMI application should be closed before disconnecting the cable.

10 Customizing

10.1 Customizing

Introduction

The Spot Options packages provides wide opportunities to customize the functionality to adapt to different types of spot weld equipment and user standards. Another purpose of the customization is to reduce the amount of data and number of variables presented to the programmer or operator.

This chapter is intended for users who do advanced customizing.

10.2 Customizing possibilities

Examples

Define the number of guns to be used. Only used guns are visible and programmable on the FlexPendant.
Define max and min values for a number of data components. The limits will be tested at runtime.
Adapt the number of data components in the Spot data types to the spot weld equipment in use.
Adapt the names of the data components in the Spot data types to the spot weld equipment in use.
Add own functionality in user routines (hooks) which are called from the kernel during the process sequence.
Change the predefined service routines for Manual Actions (ManSpot, ManCloseGun, ManOpenGun etc.)
Create new service routines for Manual Actions (MoveToHome etc.). These functions are then available from the Debug menu in the Program Editor view on the FlexPendant.
Create other equipment specific Programming instructions (for example TipDress and ChgTool).
Add equipment specific supervision and error handling. Also continuous supervision in the background.
Define the used IO signals. It is possible to have user defined names for internal used signals.
Define the MostCommon I/O list with your most frequently used signals.
Define the MostCommon instruction pick list.
Use spot data from the weld timer (for example tip force).
Change the predefined texts (xml).
Change the number of process tasks that are installed.
Configure gun arm deflection compensation in other tool directions (Software Equalizing only).
Configure 32 bit weld program groups.

10.3 Files to be changed during the customization

The customization is done by changing a number of predefined data and routines, preferably using a standard PC with RobotStudio or using an offline text editor.

The following RAPID modules and configuration files can be changed during customization:

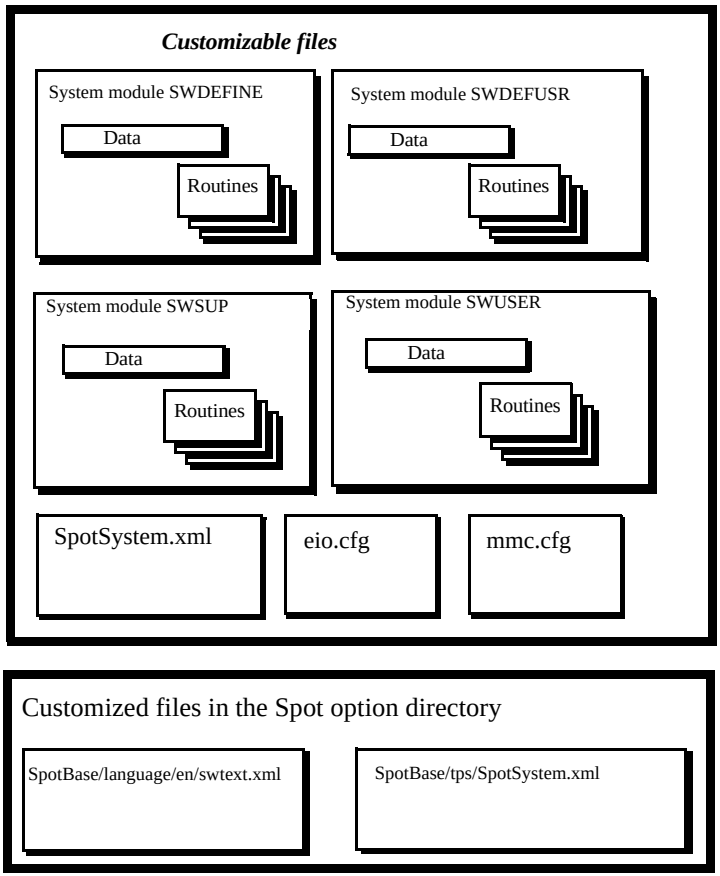


Figure 57 Files intended for customizing.

10.4 Short file descriptions

SWDEFINE (Noview)

This module is running in the Main task or in the motion tasks in a MultiMove system and contains event routines, user definition routines and routines used for manual actions, the service routines.

See *SWDEFINE* on page 160.

SWDEFUSR (Noview)

This module is intended for the person who does advanced customizing.

SWDEFUSR is running in all tasks and contains all the data definitions and setup data. It also contains the different data definition routines (for example DefineSpotData and DefineGunData).

See *SWDEFUSR* on page 166.

SWUSER (Nostepin)

This module is intended for the person who creates and tests the user program. The data and routines in this module are possible to change from the FlexPendant.

SWUSER is running in the same tasks as SWDEFUSR and contains current values for the different defined SpotWare data types. As default it also contains the different process hooks (for example SwPrepare and SwCloseGun). It is possible to move data and routines from SWUSER to SWDEFUSR and vice versa, depending on if they shall be possible to be changed from the FlexPendant and RobotStudio Online or not.

See *System modules* on page 153.

SWTEXT (.xml file)

.xml file containing text strings for the different Spot options. These texts are loaded during installation from the Spot option directory in the mediapool.

Example for strings in English:

- C:\Program Files\ABB Industrial IT\Robotics IT\Mediapool\RobotWare_5.XX.00xx\options\Spot\SpotBase\language\en

On the controller, the Mediapool is hd0a.

- for example hd0a\RobotWare_5.XX.00xx\options\Spot\SpotBase\language\en

SWSUP (Noview)

This module is intended for the person who does advanced customizations, and it contains the routines for the autonomous supervision and signal control.

SWSUP is running as a separate semistatic task, SW_SUP.

For a multi spot system there will be one SW_SUP task configured for each robot in the system.

For example:

- SW_SUP1
- SW_SUP2

See *SWSUP* on page 173.

I/O configuration (eio.cfg)

Default spot weld signals for four gun equipment are defined, and by default all signals are connected to virtual boards. The signals used must be connected to physical signals.

See *I/O configuration* on page 177.

MMC configuration (mmc.cfg)

This configuration file contains for example information about which instructions are included in the different instruction pick lists and which routines are added to the Service menu in the programming window, to be used as manual actions.

SYS configuration (sys.cfg)

It is possible to reduce the number of process tasks that are installed and configured in the system by editing the install_SpotTask.cmd script in the spot option directory.

This has to be done offline before installing the system since the RobotWare installed in the robot system is write protected.

By default four process tasks(DA_PROC1 - 4) are installed and configured after installation. Follow the instructions described below, or read the ReadMe_TaskConfig.txt in the spot option directory.

Example on a PC:

- C:\Program Files\ABB Industrial IT\Robotics IT\Mediapool\RobotWare_5.XX.00xx\options\Spot\SpotBase\install_SpotTask.cmd

1. If only one process task is required:
   - Edit the install_SpotTask.cmd and remove the lines under CONFIG_DAPROC2, CONFIG_DAPROC3 and CONFIG_DAPROC4, and reinstall the system. See example below.
2. If two process tasks are required:
   - Edit the install_SpotTask.cmd and remove the lines under CONFIG_DAPROC3 and CONFIG_DAPROC4, and reinstall the system.
3. If three process tasks are required:
   - Edit the install_SpotTask.cmd and remove the line under CONFIG_DAPROC4, and reinstall the system.

For example: Install_SpotTask.cmd for one process task:

```
# Install script for installation of the Spot tasks

setvar -var 10 -value 0
getkey -id "SpotAddOpt" -var 10 -strvar $ANSWER -errlabel
   CONFIG_DAPROC1
goto -strvar $ANSWER

#BoschCompact
config -filename $BOOTPATH/SpotBase/config/sys/
   swproc1_i.cfg -domain SYS -internal
goto -label END

#BoschBasic
goto -label CONFIG_DAPROC1

#CONFIG_DAPROC1
config -filename $BOOTPATH/SpotBase/config/sys/
   swproc1_i.cfg -domain SYS -internal

#CONFIG_DAPROC2
#config -filename $BOOTPATH/SpotBase/config/sys/
   swproc2_i.cfg -domain SYS -internal

   #CONFIG_DAPROC3
   #config -filename $BOOTPATH/SpotBase/config/sys/
```

```
      swproc3_i.cfg -domain SYS -internal

   #CONFIG_DAPROC4
      #config -filename $BOOTPATH/SpotBase/config/sys/
      swproc4_i.cfg -domain SYS -internal

   #END
```

Note!

Changes in the install_SpotTask.cmd script requires that the system is reinstalled (I-start).

10.5 How to change the number of guns to be used

As default it is possible to use four guns but it is possible to use and setup data for up to ten guns in the system. Use a text editor to edit the Spot user modules.

	Action	Note
1	Change the data SW_NOF_GUNS in SWDEFUSR to desired value (1-10).	CONST num SW_NOF_GUNS := 10;
2	Add or remove signals in eio.cfg for all equipment to be used in a similar manner as the predefined signals for gun equipment 1 to 4.	
3	Increase the number of data in following arrays in SWUSER (The size shall be in agreement with the used number of guns):	• curr_gundata • curr_spotdata • curr_forcedata • man_forcedata
4	Increase the routines *SwPowerOn* in SWDEFINE and *SwInitUserIO* in SWUSER, and *DefSupIO* in SWSUP with `SignalC*` instructions for the signals corresponding to the extra gun equipment.	`*SignalConnect` is an internally used instruction with the same function as `AliasIO`.

Note!

Any code changes in SWUSER, SWDEFUSR, and SWSUP modules requires a P-Start to take effect.

10.6 How to define max/min values for a number of data components

It is possible to define max and min values for a number of data components. The limits will be tested at runtime. Change following data in SWDEFUSR to desired values:

Data	Description	Default value
MAX_PRE_CLOSE_T	Max value for pre_close_time in gundata [s]	0.5
MAX_PRE_EQU_T	Max value for pre_equ_time in gundata [s]	0.1
MAX_TIP_FORCE	Max value for tip_force in spotdata and forcedata [N]. For a pneumatic gun the default max value is 7 force levels, I/O group.	6000 or 7
MAX_PLATE_THICK	Max value for plate_thickness in spotdata and forcedata [mm], (Servo guns only)	40
MAX_PLATE_TOLER	Max value for plate_tolerance in spotdata and forcedata [mm], (Servo guns only)	1
MAX_PROG_NO	Max value for prog_no in spotdata	255
MIN_WELD_TOUT	Min value for weld_timeout in gundata [s]	2
MAX_WELD_TOUT	Max value for weld_timeout in gundata [s]	10
MIN_FORCE_T	Min value for force_time in forcedata [s]	0.05
MAX_FORCE_T	Max value for force_time in forcedata [s]	10
MAX_SIM_T	Max value for sime_time in simdata [s]	10
MAX_REL_DIST	Max value for release distance in gundata. (Spot equalizing only)	15
MAX_DEFLECTION	Max value for deflection distance in gundata. (Spot equalizing only)	15
MAX_TOUCHUP_STEP	Maximum allowed step value (mm) for weld position touch up.	10
MIN_TOUCHUP_STEP	Minimum allowed step value (mm) for weld position touch up.	0.1

10.7 How to change the Spot data types

	Action	Note
1	Change the definition of the Spot data types in SWDEFUSR to desired.	It is possible to • Add or delete data components. • Move data components from for example gundata to spotdata. • Change the names of the data components.
2	Change corresponding instructions in the data definition routines in SWDEFUSR. These routines are used to connect the user defined data components to internal data.	• DefineSpotData • DefineGunData • DefineForceData • DefineSimData
3	Change the structure and the default values of following arrays in SWUSER (if corresponding data type is changed):	• curr_gundata • curr_spotdata • curr_simdata • curr_forcedata • man_forcedata

Text strings can be changed in the swtext.xml file if necessary. This file is located in the Spot option directory in the RobotWare mediapool.

10.8 How to add functionality in the process sequence

	Action	Note
1	Add code to the process hooks in SWUSER	See description of the process hooks in *System modules* on page 153.

10.9 How to change, add or delete Manual Actions

	Action
1	Change the routines for the Manual Actions. This routines (ManSpot, ManCloseGun, ManOpenGun etc.) are found in module SWDEFINE.
2	If Manual Actions are added or deleted, or if Manual Action names are changed then also the MMC configuration has to be changed for the routine to be visible in the view service routines menu. Change the names under MMC_SERV_ROUT_STRUCT:

10.10 How to create other equipment specific progr. instructions

	Action
1	Create global routines with desired name, syntax and functionality. Use a system module with the attribute NOVIEW or NOSTEPIN.
2	The MMC configuration has to be changed to define the instruction syntax and to get the new instructions in an instruction pick list.

10.11 How to add equipment specific supervision and error handling.

	Action
1	If the supervision during the weld process has to be changed, add code to the process hooks in SWUSER. See description of the process hooks in *System modules* on page 153
2	If the autonomous supervision has to be changed, the code in the supervision module SWSUP has to be changed. The normal way to add supervisions is to connect the supervised signal to a trap routine (SupTrap) and create a new supervision routine which is called from the trap routine. See the predefined supervisions in SWSUP.

Note!

All changes in this module requires a P-Start to be activated.

10.12 How to use own signal names on internal used signals

Some weld timers are prepared for storing data like tip_force and plate_thickness for each weld program in the timer. When the robot controller sends a new program number the timer responds with this data (for example on separate input groups). Then it is possible to use this data instead of corresponding data from current spotdata

	Action	Note
1	The following routine calls in SWDEFINE are used to inform the kernel about current signal names:	• DefGunIO • DefInternalIO • DefTimerIO
2	This routine call in SWUSER is used to define the signals used in all process hooks	• SwInitUserIO
3	This routine call in SWSUP is used to define the supervision signals used for all gun equipment.	• DefSupIO
4	Change the parameters in these instructions so they are in agreement with the used signal names in the EIO.cfg.	
5	Update the SpotSystem.xml file with the used signal names.	See example below.

The Spot MMI, FlexPendant Interface uses a configuration file with the signal names defined, this file needs to be edited if other names than default are used.

Open the *SpotSystem.xml* file in a text editor and edit this file according to the instructions described inside, this file is copied to the Spot directory in the system together with the customized rapid modules.

Tip!

If a customized copy of SpotSystem.xml is placed on the /Home/Spot directory this file will be used instead of the default file in the option directory after the system is restarted, it will also be stored in a backup.

Note!

If the I/O signal names listed in SpotSystem.xml does not match the actual I/O configuration the Spot MMI, FlexPendant Interface will not work correctly.

10.13 How to use spot data programmed in the weld timer

Some weld timers are prepared for storing data like tip_force and plate_thickness for each weld program in the timer. When the robot controller sends a new program number the timer responds with this data (for example on separate input groups). Then it is possible to use this data instead of corresponding data from current spotdata.

	Action	Note

1	Add code in SwClose-Gun in SWUSER to transfer the weld timer force data to the internal data.	E.g
		`VAR num tip_force;`
		`!Read force from timer.`
		`tip_force := GInput (gi_timer_force);`
		`!Update the internally used force data`
		`int_spotdata{GunNum}.tip_force := tip_force;`

2	Remove not used data components from spot-data in SWDEFUSR. Do not forget to modify the data declarations in the SWDEFUSR and SWUSER modules. See *How to change the Spot data types* on page 246	E.g remove the tip_force parameter.
		Definition of process spot data
		`RECORD spotdata`
		`num prog_no;`
		`num tip_force;`
		`num plate_thickness;`
		`num plate_tolerance;`
		`ENDRECORD`
		Definition of process spot data
		`RECORD spotdata`
		`num prog_no;`
		`num plate_thickness;`
		`num plate_tolerance;`
		`ENDRECORD`

10.14 How to hide the mechanical gun equalize function

As default this function is activated but if the used gun not is equipped with equalizing possibilities it is possible to hide this function.

	Action
1	Change the data manage_equalize in SWDEFUSR to FALSE.
2	Remove the equalize signal in eio.cfg for all equipment.
3	Remove or rename the pre_equ_time from gundata, see *How to change the Spot data types* on page 246.
4	Remove the optional argument DOEqualize from DefGunIO instructions in SWDEFINE. **E.g** **SwDefGunIO 1 \DOEqualize := g1_equalize \DIForceComplete := force_complete;** **SwDefGunIO 1 \DIForceComplete := force_complete;**

10.15 How to set the number of automatic rewelds after weld error

As default the automatic reweld function is deactivated.

	Action	Note
1	Set the data auto_reweld in SWDEFUSR to desired value.	for example 0 - deactivated, no automatic reweld 1 - one extra reweld 2 - two extra rewelds etc.

10.16 How to configure arm deflection compensation in other directions

As default gun arm deflection compensation in the tool z-direction is activated. But for some specific guns there may be a need to have arm deflection also in other directions, for instance in the x-direction.

The deflection direction for x and y can be configured by specifying a positive or negative deflection value for these parameters.

Follow these steps to configure arm deflection also in the tool x- and/or y-directions.

	Action

1	Add new parameters for the deflection in x and y direction in the gundata record in the user module SWDEFUSR.SYS. Also rename the deflection_dist to deflextion_dist_z.
	For example, definition of process gun data
	`RECORD gundata`
	`...;`
	`num deflection_dist_x;`
	`num deflection_dist_y;`
	`num deflection_dist_z;`
	`num deflection_force;`
	`...;`
	`ENDRECORD`

	Action

2	Add the parameters for the deflection in x and y direction in the DefineGunData procedure in the SWDEFUSR.SYS module.
	This procedure will update the internally used gundata when run.
	For example, called just before execution of SwPowerOn, SwStart or SwReStart.
	`PROC DefineGunData ()`
	`FOR i FROM 1 TO SW_NOF_GUNS DO`
	`...;`
	`int_gundata{i}.deflection_dist :=`
	`curr_gundata{i}.deflection_dist_z;`
	`int_gundata{i}.deflection_dist_x :=`
	`curr_gundata{i}.deflection_dist_x;`
	`int_gundata{i}.deflection_dist_y :=`
	`curr_gundata{i}.deflection_dist_y;`
	`int_gundata{i}.deflection_force :=`
	`curr_gundata{i}.deflection_force;`
	`...;`
	`ENDFOR`
	
	`ENDPROC`

3	Add the init values for the new deflection values in the curr_gundata structure in the user module SWUSER.SYS.
	For example, gundata for each gun
	`PERS gundata curr_gundata{SW_NOF_GUNS}:=`
	`[["SERVOGUN",0.05,0,0,1000,0,10,2,TRUE,`
	`0,0,10,5,3,1,3000,0.1,50],`
	`["NOT USED",0.05,0,0,1000,0,10,2,TRUE,`
	`0,0,10,0,0,0,5000,0.1,-1],`
	`["NOT USED",0.05,0,0,1000,0,10,2,TRUE,`
	`0,0,10,0,0,0,5000,0.1,-1],`
	`["NOT USED",0.05,0,0,1000,0,10,2,TRUE,`
	`0,0,10,0,0,0,5000,0.1,-1]];`

4	Upload the customized SWDEFUSR.SYS module to the /Home/Spot directory on the system and perform a p-start to reload all RAPID modules.

10.17 How to use dnum weld program numbers (32 bit signal group)

As default the weld program parameter in the `spotdata` data type is of the type `num`. This is good enough in most cases, but if there is a need to use weld program numbers larger than the numeric data type limit of 8388608, the data component `prog_no` in `spotdata` can be exchanged to a `dnum` type which will allow the use of weld program signal groups up to 32 bits (4294967295).

Follow these steps to change the *prog_no* data component in spotdata to a dnum data type. All changes needed here are done in the SWDEFUSR.SYS user module.

	Action

1	Change the existing weld program parameter in the spotdata record from a num to a dnum type in the SWDEFUSR.SYS module.
	For example, definition of process spot data
	`RECORD spotdata`
	`! num prog_no; ! Old data`
	`dnum prog_no; ! New data`
	`num tip_force;`
	`num plate_thickness;`
	`num plate_tolerance;`
	`ENDRECORD`

2	Remove the old data assignment to the *int_spotdata.prog_no* and replace it with the new data *int_spotdata.program_number* in the DefinSpotData routine in SWDE-FUSR.SYS.
	This procedure will update the internally used spotdata when run.
	For example, called in the beginnning of each Spot shell routine.
	`PROC DefineSpotData (spotdata Spot, num GunNum)`
	`! int_spotdata{GunNum}.prog_no := Spot.prog_no; ! Old data`
	`int_spotdata{GunNum}.program_number := Spot.prog_no; ! Assign to the new data`
	`int_spotdata{GunNum}.tip_force := Spot.tip_force;`
	`ENDPROC`

	Action

3	Remove the old data assignment to *curr_spotdata* and replace with the new data in the UpdateSpotData routine in SWDEFUSR..SYS
	This procedure will update the *curr_spotdata* when run.
	For example, update curr_spotdata.
	`! Called at the start of motion`
	`PROC UpdateSpotData (z_int_spotdata Spot, num GunNum)`
	`! curr_spotdata{GunNum}.prog_no := Spot.prog_no; !Old data`
	`curr_spotdata{GunNum}.prog_no := Spot.program_number; !New data`
	`curr_spotdata{GunNum}.tip_force := Spot.tip_force;`
	`ENDPROC`

4	Upload the customized SWDEFUSR.SYS module to the /Home/Spot directory on the system and perform a p-start to reload all rapid modules.

Note!

Do not forget to change the size of the actual weld program signal group in the I/O configuration, for instance the *g1_weld_prog* signal group needs to be increased if the data is changed.

10.18 How to package/install the result from the customization

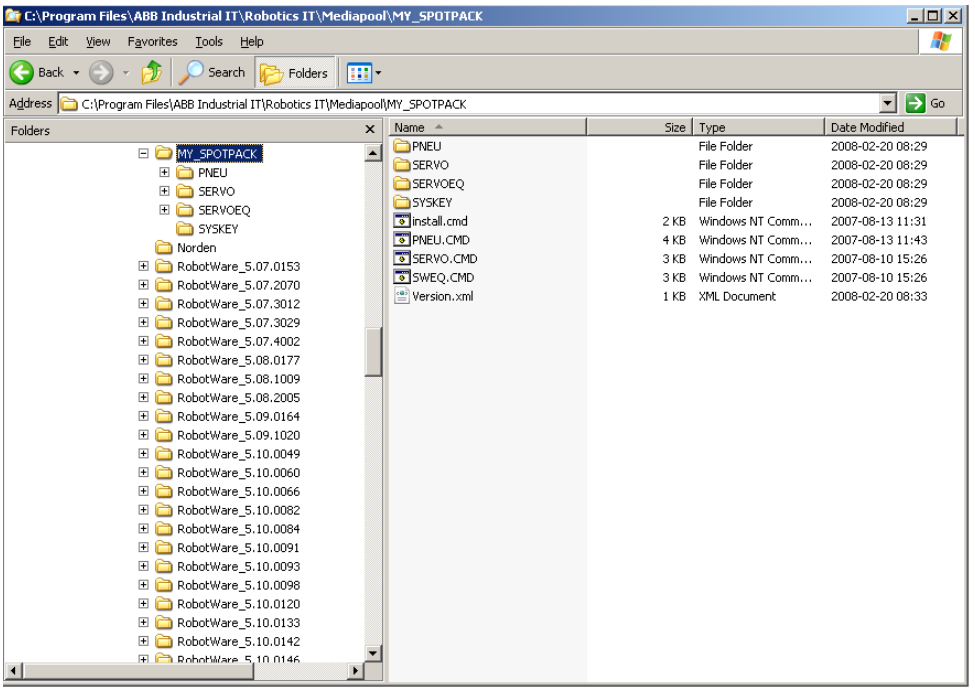
Packaging

After customizing, it is a good idea to create a new directory with the changed files for each customized variant and load it as an additional option.

This additional option can then be included via the system builder in RobotStudio.

When the additional option is loaded the default system modules loaded by the installation are replaced by the customized modules and the influenced cfg files are replaced by the customized versions for desired variant, for example eio.cfg, mmc.cfg and moc.cfg.

This additional option should be located in the MediaPool.



Contact us

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

3HAC024762-001 Rev H, en