CONTENTS

Page

Page

Page

Page

1 SpotWare for Servo Guns - Summary

The Spotweld options are general and flexible software platforms for the creation of **customised** and **easy to use** function packages for different types of spotweld systems and process equipment.

There are two different SpotWare options supporting spotweld with servo guns.

The **SpotWare Servo** package is used for sequential welding with one or several servo gun equipments.

The **SpotWare Servo Plus** package provides support for sequential welding with one or several servo gun equipments, but also welding with two servo guns at the same time.

All SpotWare options provide 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 carried out by means of digital inputs and outputs.

It should be noted that the SpotWare options are general and can be extensively customised. They have a default “ready to use” functionality directly after install but it is intended that some configuration data, RAPID data and RAPID routines should be changed during customising.

1.1 Spot welding features

The SpotWare Servo and SpotWare Servo Plus package contain the following features:

- Handling of up to eight servo guns in sequence.
- Welding with two guns at the same time (if SpotWare Servo Plus is installed).
- Fast and accurate positioning
- Gun pre-closing
- Gun equalising
- 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.
- Wide customising possibilities.

- Default “ready to use” functionality directly after install.
- Constant tip force during welding.
- Detecting of missing or improper plates.
- Gun calibration functions.
- Spot counters and tip wear data for each used gun.
- Fast switching between different servo guns with a tool changer. Note: This feature requires the Servo Tool Change option.

1.2 Principles of SpotWare Servo and SpotWare Servo Plus

This SpotWare options are based on the DAP (Discrete APplication) concept with separate handling of motion, spot welding and continuous supervision. On its way towards the programmed position, the motion task will trigger actions in the spot-welding tasks.

The tasks work with their own internal encapsulated variables and with persistent data which is fully transparent for all tasks.

For well defined entries, calls to user routines offer adaptations to the plant environment. A number of predefined parameters are also available to shape the behaviour of the SpotWare instructions.

A program stop will only stop the user program execution. The process and supervision carry on their tasks until they come to a well defined process stop. For example, the process will carry on and finish the weld and open the gun, although the program has been stopped during the weld phase.

1.3 Programming principles

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* must be used. These instructions are available if SpotWare Servo Plus is installed.

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 customising purposes, e.g. adaptations for a specific process equipment.
- The system module SWUSER: RAPID routines and global data for changing process and test behaviour.
- System parameters: the I/O Signal configuration and the Manipulator configuration. See User’s Guide - System Parameters

1.4 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 weld at the end position.
SpotJ	Control the motion, gun closure/opening and the welding process. Move the TCP along a non-linear path and perform a spot weld at the end position.
SpotML	Control the motion, gun closure/opening and 1 - 2 welding processes. Move the TCP along a linear path and perform spot welding with 1 - 2 gun equipments at the end position. Only available if SpotWare Servo Plus is installed.
SpotMJ	Control the motion, gun closure/opening and 1 - 2 welding processes. Move the TCP along a non-linear path and perform spot welding with 1 - 2 gun equipments at the end position. Only available if SpotWare Servo Plus is installed.
SetForce	Close the gun for a predefined time then open the gun.
CalibL	Calibrate the gun during linear movement to the programmed position.
CalibJ	Calibrate the gun during non-linear movement to the programmed position.
Calibrate	Calibrate the gun in the current position without movement.
TuneGun	Tune motion parameters for the servo gun.
TuneGunReset	Reset tuned motion parameters for the servo gun.

1.5 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

1 Servo gun motion control

Servo gun introduction

External axes

The robot controller has additional functionality to control external axes configured as servo guns (other types of supported external 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 internal drive modules. Internal drive modules are mounted either inside the robot cabinet (i.e IRB6400R + one servo gun) or in a small separate cabinet, a *Distributed Drive Unit*, (i.e IRB6400R + two stationary servo guns, one of the guns is controlled by a drive module inside the robot cabinet and the other one is controlled by a DDU).

Motion servo gun parameters

A motion servo gun parameter file is installed in the controller for each servo gun. The parameter files are optimally designed as regards system behaviour and motion/ process performance.

To be able to read and change most of the parameters from the teach pendant unit after installation, the system must be booted in motion service mode.

With *SpotWare Servo* some gun specific system parameters may be updated temporarily directly in the robot program using the instruction TuneGun. This function makes tuning of gun parameters easier.

General motion control for servo guns

The motion functionality described in this section is common for servo guns and most other types of external axes. The description is however adapted for servo guns.

Activation/Deactivation

A servo gun may be activated, when the robot and all external 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 is not possible to activate another gun until the first one is deactivated (mutual exclusion).

Deactivation of the gun is only needed when the gun must be disconnected, for service or for a tool change. The deactivation stores the gun's current position. This position is restored when the gun is activated next time. Deactivation is performed with a DeactUnit instruction and this also stops the control and monitoring of the axis.

Jogging

The position of the gun arm can be jogged with the joy stick (see User's Guide, *Jogging external axes*). The distance between the two tips is displayed in the jogging window, expressed in mm. An out of range supervision stops the movement if the gun is reaching its max. or min. stroke. Min. stroke is normally zero or a small negative value (gun tips closed to contact with each other).

Coordinated movements of robot and servo gun

A servo gun axis may be coordinated or not coordinated with the robot movements. If it is not coordinated, it moves independently of the robot movements, e.g. when jogging the axis in teach mode or when closing the gun tips with a force, only the servo gun axis moves. However during normal movements (i.e. MoveL, MoveJ, MoveC) in program execution, the gun axis movement is synchronised to the robot movement, in such a way that both movements are completed exactly at the same time. The combined path of robot and servo gun(s) is repeatable and independent of the programmed speed. The robot TCP path is the same, irrespective of the programmed movements of the servo gun's movable arm.

A robtarget includes positional data for external axes which are also set when a ModPos is performed. Example:

p10 is a robtarget RAPID data.

p10.extax.eax_a	The position of the external axis with *logical axis* 7.
p10.extax.eax_b	The position of the external axis with *logical axis* 8.
...	
p10.extax.eax_f	The position of the external axis with *logical axis* 12.

Logical axis is a system parameter defined for each axis (Topics: Manipulator, Type: JOINT). The robot itself uses logical axes 1-6 and external 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 which are not activated.

Supervision during general motion control

An out of range supervision stops 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 detects whether the gun arm collides or get stuck. A motion error occurs and the motion is stopped.

Asynchronous movements with force control

The motion functionality described in this section is only valid for servo gun axes and includes opening and closing of the gun to a programmed force and plate thickness.

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 start to run the gun arm immediately to the expected contact position (plate thickness). The closing movement interrupts an on-going coordinated movement (synchronous movement) of the gun. When the tips reach 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 is applied to achieve the desired tip force.

The force remains constant until an opening is ordered.

Opening of the gun reduces the tip force to zero and move the gun arm back to the pre-close position.

A gun opening synchronisation error *may* occur if the robot is moving during the opening movement. A motion error occurs, “Gun opening could not synchronise with robot movement”.

A gun opening will however always work during a robot movement as long as this movement does not include a coordinated movement of the gun. Make sure that the programmed servo gun position is unchanged in those consecutive robtargets that are executed during gun opening.

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 is 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 *SpotWare Servo*, the closing, opening, thickness measurement, weld start and opening are all integrated in the SpotL/SpotJ instruction.

Squeezing without welding

A gun closing is also typically done when carrying out tip dressing or when changing tips. The force is held constant for a certain time and then the gun is opened up again.

In *SpotWare Servo*, the SetForce instruction squeezes the gun with a specified force, thickness and for a specified time. SetForce takes a forcedata as argument where these values are setup. A thickness test is integrated in the instruction.

Tip calibration

Tip calibration includes typically two consecutive gun closings and openings. See the *tip management* section.

Supervision during asynchronous movements with force control

During the position control phase of the closing/opening, a motion supervision is active for the servo gun to detect whether the arm collides or gets stuck. A motion error is then triggered.

There is a maximum motor torque defined in the motion parameters for the gun that is never exceeded in order to protect the gun from damage. If the force is programmed out of range, according to the gun's force-torque table, the output force is limited to this maximum allowed motor torque and a motion warning is logged.

During the force control phase, the motion supervision supervises the gun position not to exceed a certain distance from the expected contact position. This distance, *fc position limit*, is setup in the motion gun parameters (Topics: Manipulator, Type: Supervision) and is typically dependent on the flexibility of the gun arm. This supervision protects the gun if for instance one tip is lost.

During the force control phase, speed limitation is also active, which limits the speed of the gun. The speed limit value is defined in the gun parameters (see the tuning chapter in *External Axes* manual), and this limitation gives a controlled behaviour of the gun if the gun is ordered to close to a position where the tips are not in contact. There is also an active supervision to ensure that the actual speed does not exceed this limit too much (*fc speed limit factor*).

Tip management

The tip management functionality finds and calibrates the contact position of the gun

tips automatically. It also updates and monitor the total tip wear of the gun tips. The total tip wear is stored in a RAPID gundata.

The tips are calibrated with special RAPID instructions. Typically, two gun closings are performed during a calibration. The calibration may be done when the robot is standing still (SpotWare Servo instruction: *Calibrate*), or during a robot movement (SpotWare Servo instructions: *CalibL, CalibJ*).

Three different types of calibration are supported: *tip wear*, *tip change,* and *tool change*. All three calibrate the contact position of the tips. The total tip wear is however updated differently by these methods:

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 takes 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 can 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 can handle big positional adjustments of the gun.

The method should always be used after reconnecting a gun since the activation restores the latest known position of the gun and that position may be different from the actual gun position; the gun arm may have been moved when disconnected. This

calibration method can 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 centre point adjustment

Half the total tip wear of the gun may be used to adjust / optimise the tool centre point of the robot tool (RAPID *tooldata*).

Supervision during tip calibration

The same supervision is active during calibration as during *asynchronous movements with force control*.

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 the service menu 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).

Service calibration

After installing the gun parameters and restarting the system, the initial gun position must be synchronised. Jog the gun to contact position and fine calibrate the axis. Once synchronised, further updating of the gun arm position should be done with RAPID tip calibration commands.

Initial tip calibration

Run a *tip change* calibration. Then run a *tip wear* calibration. These two calibrations initialise the position, tip wear, and thickness detection functionality.

Force calibration

There is a Rapid service routine to calibrate the motor torque vs tip force characteristics. 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. The system is restarted to finish the calibration procedure in order to activate the new calibration.

Disconnecting/Reconnecting a servo gun

If the servo gun is deactivated, using the DeactUnit instruction, it may be disconnected and removed. The gun position at deactivation is restored when the gun is connected and reactivated. Make a tool change calibration to ensure the tip position is OK.

Replacing a servo gun

Install the parameters for the spare gun in the service menu using ‘add and replace parameters’. The spare gun must have the same parameter names as the original gun, otherwise the installation just adds the new gun, keeping the old gun in parallel. If the spare gun has identical parameters, it may run directly without installing new parameters. Always make an *initial tip calibration* (see above) after connecting the gun in order to update the position and the tip wear.

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.

Servo tool change

It is possible to switch servo gun during production using a tool changer. The functionality is realised as the option Servo Tool Change. The gun parameters for each gun are installed in parallel but only one of the mechanical units can be activated at the same time. The motion parameters for these guns are configured to share the same drive and measurement system.

Up to 8 external 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.

Example, tool change procedure

The procedure to switch between gun 7A and gun 7B must include these minimal actions (excluded here is the communication needed with the tool changer, the tool

stand and necessary robot movements):

1 Deactivate gun 7A. (The position of gun 7A is stored)

2 Disconnect gun 7A. (Disconnect the servo gun motor cables)

3 Connect gun 7B. (Connect the motor cables to motor 7B)

4 Activate gun 7B. (The latest position of gun 7B is restored)

5 Run a tool change tip calibration of gun 7B. (Make sure the position is correct)

Warning 1: If the servo gun axis has been moved during deactivation, the position of the axis may be wrong after activation, and this is not detected by the controller. The position after activation is 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 adjusts any positional error caused by gun movements during deactivation.

Warning 2: It is important that no other mechanical units used with one tool changer are activated, except the one corresponding to the currently connected servo gun! An activation of the wrong mechanical unit may cause unexpected movements or errors. Some tool changers support IO signals that specify which gun is currently connected. This information may be used to ensure that the correct mechanical unit is activated. It is also possible to lock activation of mechanical units that are not connected by specifying a digital input (DI) in the *connection relay*, which is a motion system parameter (Topics: Manipulator, TYPE: relay) for each servo gun. This digital input, which also is setup in the EIO.cfg, is read when the mechanical unit is being activated. If set to 1 the activation takes place normally, otherwise a recoverable motion error occurs and the activation is denied.

Other references

SpotWare Servo Manual	CalibL, CalibJ, Calibrate, SpotL, SpotJ, SetForce TuneGun, gundata, spotdata, forcedata
User’s Guide	General motion control and programming
Rapid Reference Manual	ActUnit, DeactUnit, MoveL, MoveJ, robtarget, tooldata
External Axes	Hardware: motors, resolvers, drives etc. Servo gun parameters, tuning a servogun.

1 Programming

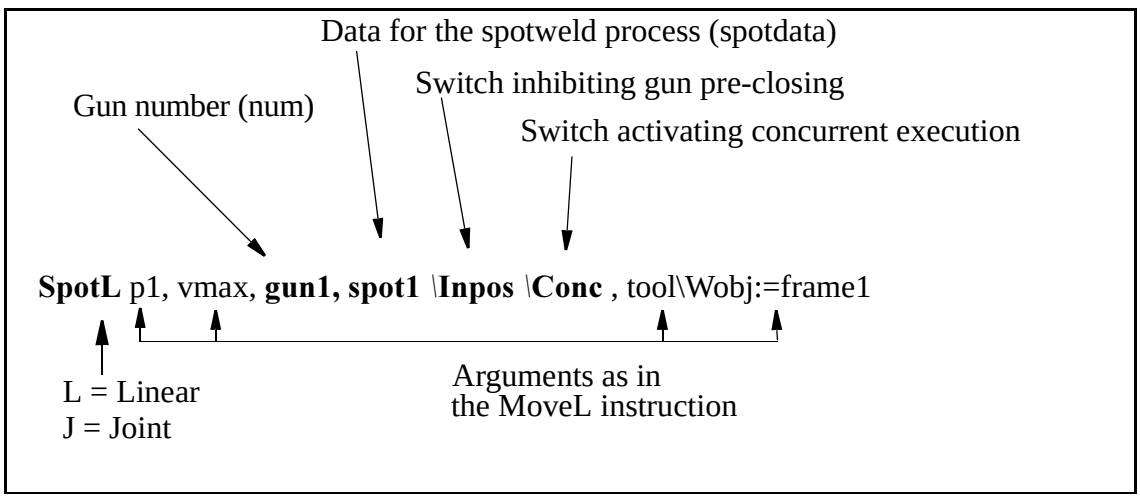
This chapter describes the basic functions and steps to take when creating, testing and running spot weld programs with SpotWare Servo or SpotWare Servo Plus.

It is assumed that the servo gun is installed and calibrated. See SpotWare Servo - *Servo gun motion control*.

The spotweld instructions for sequential welding

SpotL and *SpotJ* are the basic spotweld instructions in the SpotWare Servo package. The instructions includes a movement to the weld position and performing the desired weld process. They basically contain 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 SpotWare Servo - *SpotL/J*



Defining spotweld data

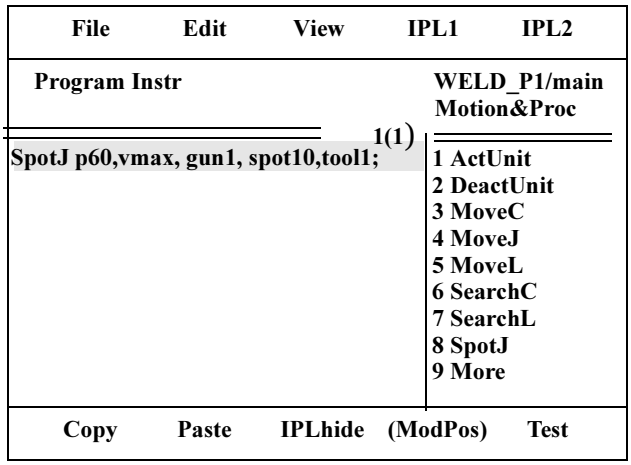
Before starting to program the instructions, you should define the spotweld data that is to be used. This data is divided into two types:

- *spotdata*; describes the spotweld process specific data for a specific spot. For further details, see SpotWare Servo - *spotdata*.
- *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. For further details, see SpotWare Servo - *gundata*.

Programming spotweld instructions

- Jog the robot to the desired destination position and jog also the gun axis to the desired preclose tip position.
- Call up the instruction pick list by choosing **IPL1: Motion&Process**.
- 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 spotweld instruction.



- Change the arguments if necessary.

Other spotweld instructions are programmed in a similar way.

Programming example 1

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

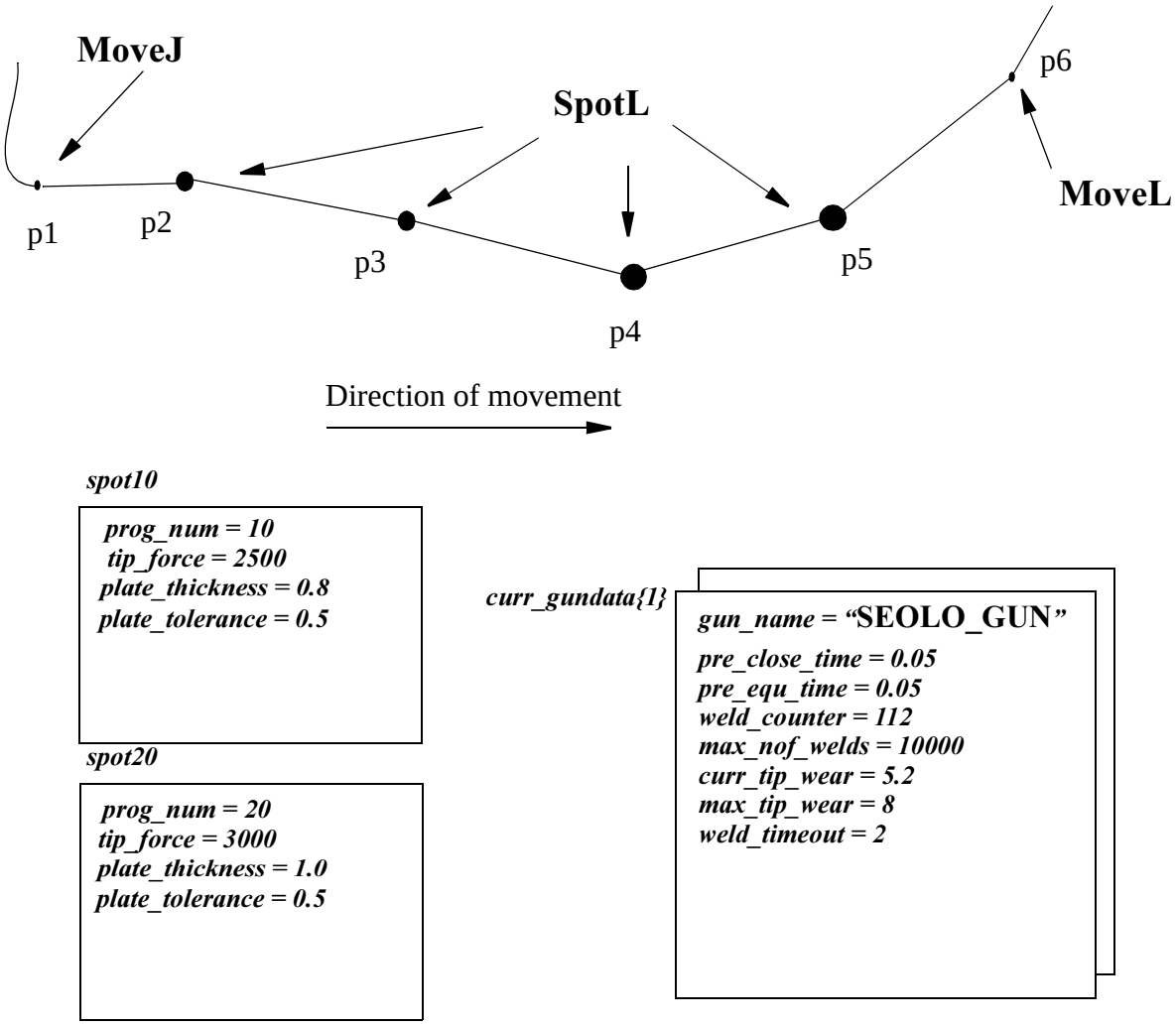


Figure1 Programming example.

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;
```

Editing spotweld instructions

Change current spotdata:

- Mark current spotdata in the instruction.
- Call up the data by choosing **Edit: Value.**
- Change desired value.
- Press OK.

Change to another spotdata:

- Mark current spotdata in the instruction.
- Press Enter.
- Select desired spotdata from the list.
- Press OK.

Testing spotweld instructions with simulated welding

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

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. This is done by setting *inhib_close* or *no_plates* to TRUE in curr_simdata.

Gun preclosing

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

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

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

Note. The preclosing can be disabled by choosing the \InPos argument in the instruction.

Gun equalising

The spotweld instructions have a function for equalising the gun, i.e. when approaching the position a signal is activated to be used for the equalising. The signal is deactivated after the weld process before the next robot motion is released.

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

Tip dressing

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 the next time to do tip dressing or tip exchange.

Manual actions

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

• Choose **Special: Call Service Routine**

The following Service Routines are predefined for manual actions:

ManCloseGun	Close the gun according to data in man_forcedata.
ManOpenGun	Open the gun.
ManSpot	Perform a weld in current position according to data in curr_spotdata.
ManSetForce	Perform a SetForce action according to data in man_forcedata.
ManCalib	Perform a calibration of the gun.
ManForceCalib	Perform a force calibration of the gun.

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

Running spotweld instructions in concurrent execution

To save some cycle time the spotweld instructions can be running in concurrent execution, i.e. the program execution is continuing while welding is in progress, and it will not stop until the next motion instruction. This concurrent execution is activated if the \Conc switch is used in the instruction.

Therefore, if the running is stopped during the weld process, the program pointer is already moved to the next spot or motion instruction. It is important to remember this fact when modifying spotweld positions with ***ModPos***.

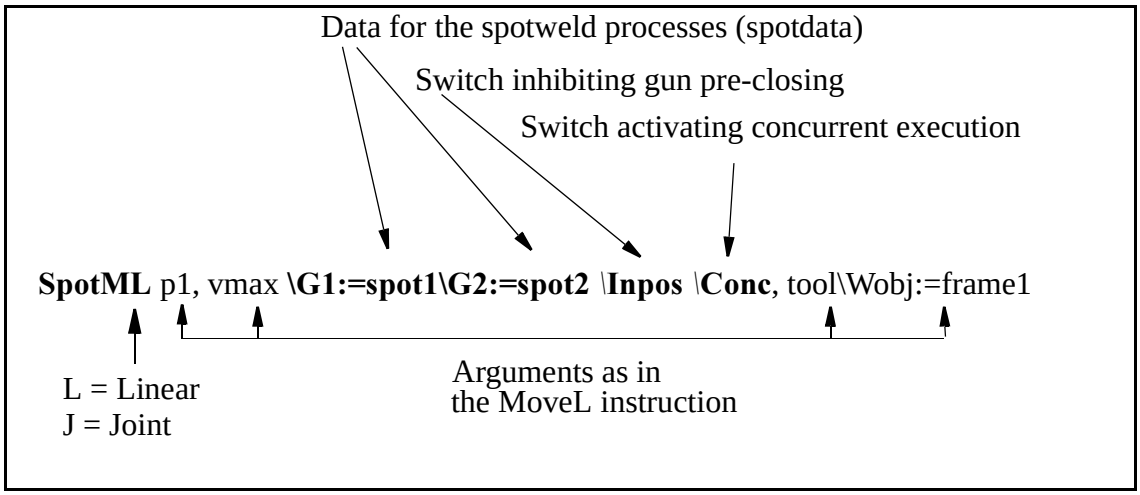
If the \Conc switch is used it is recommended to execute the program in step-by-step mode when positions are modified. In this mode the program pointer always corresponds to the robot position.

Spotweld instructions for simultaneously welding with multiple guns

SpotML and SpotMJ must be used if welding with several guns at the same time is desired. For servo guns it is possible to use two 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.

These instructions are only available if the SpotWare Servo Plus option is used.

For further details, see SpotWare Servo - *SpotML/J*



Programming example 2

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 types of spotdata, *spot10* and *spot20*. Current gun parameters have been set up in the first and second gundata in the curr_gundata array in SWUSER.

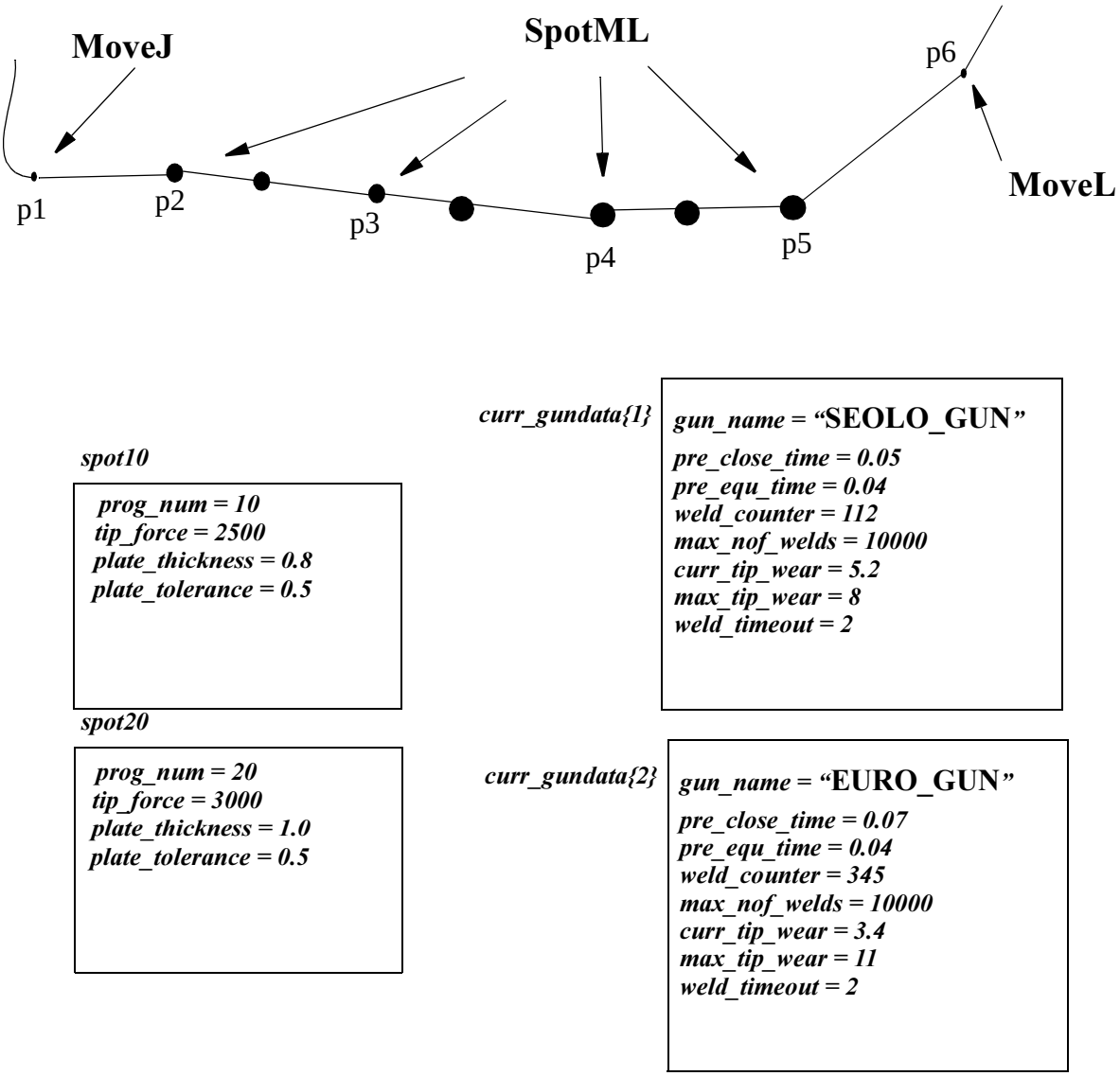


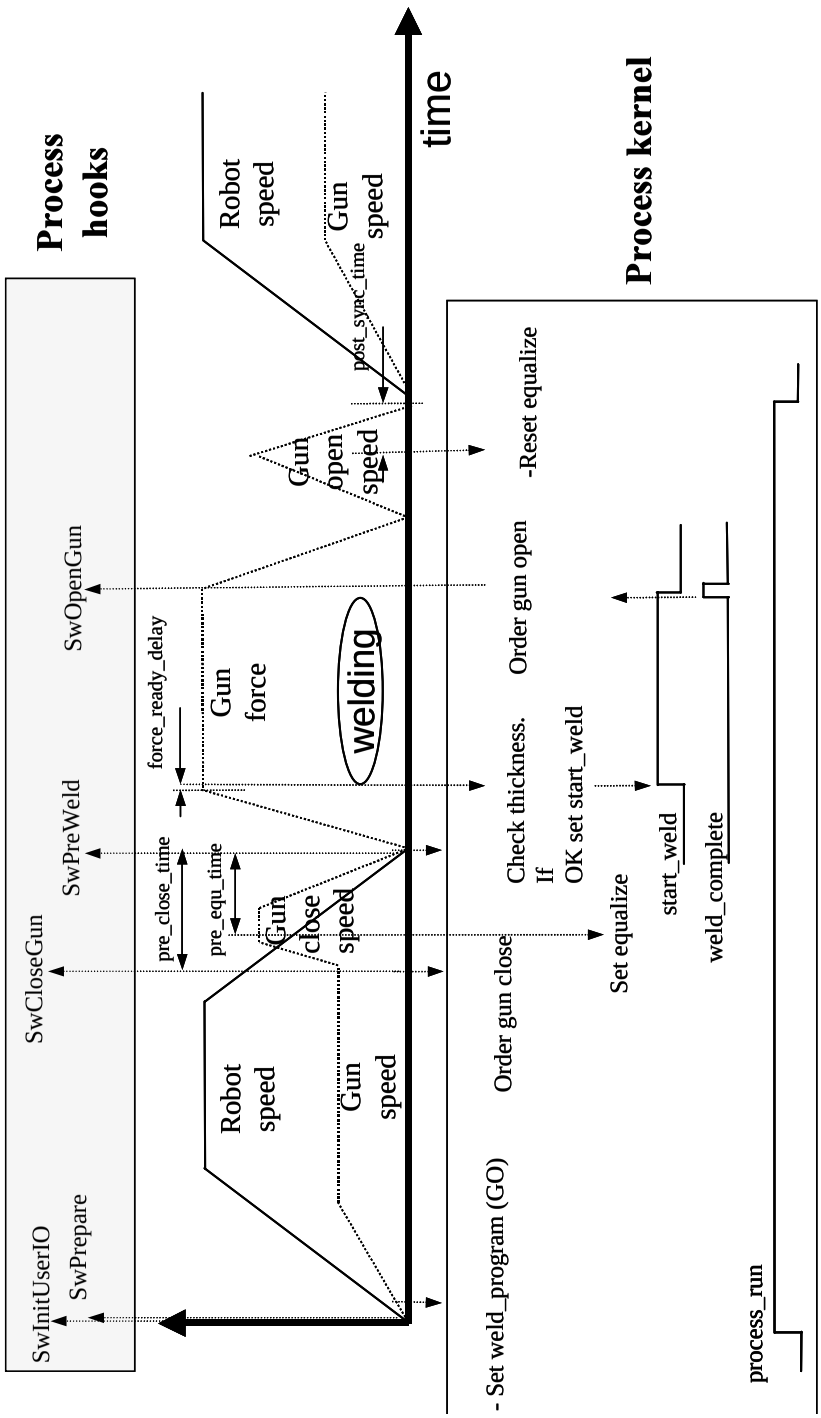
Figure2 Programming example.

RAPID code sequence:

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

Weld process timing

The figure on this page shows the weld process timing and where in the sequence the user hooks will affect the internal behaviour. If several gun equipments are used at the same time they are handled in different tasks so each process is independent.



1 SpotL/SpotJ The basic Spot Welding instructions

SpotL and *SpotJ* are used in spotwelding when welding with one gun or several guns in sequence. The instructions are used to control the complete welding sequences, i.e. 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.

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.

Spotdata spot10 contains spot weld specific parameters for the spot in p100, e.g. desired program number and pressure.

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

Arguments

SpotL ToPoint Speed GunNo Spot[\InPos] [\Conc] Tool [\WObj]

SpotJ ToPoint Speed GunNo Spot[\InPos] [\Conc] Tool [\WObj]

ToPoint Data type: *robtarget*

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

Speed Data type: *speeddata*

The speed data that applies to movements. Speed data defines the velocity for the tool centre point, the tool reorientation and external 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.

[\Conc] Data type: *switch*

Normally (without \Conc) the program execution continues to the next instruction when the spot weld process is ready.

With the optional argument \Conc the program execution continues to the next instruction directly after the weld has started and is not blocked until the next order that contains a robot motion.This will under some circumstances save cycle time. But be careful, the program pointer is moved to the next instruction and all logical instructions before the next movement instruction are executed during the weld process.

Tool Data type: *tooldata*

The tool in use when the robot moves. The tool centre 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 external axes are used, this argument must be specified in order to perform a linear movement relative to the work object.

Communication

SpotL/J communicates with the weld equipment by using parallel signals.

For a complete description of the I/O configuration, see System Parameters - *Spot Welding*.

Program execution

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

- The robot and gun start to move towards the programmed position.
- **SwPrepare** is executed. (Preweld supervision.)
- The weld program number is set.
- The gun starts to close before the position is reached (unless argument \InPos is used), according to the predefined preclosing time.
- **SwCloseGun** is executed.
- Equalising is activated.
- Current spotdata (curr_spotdata in SWUSER) is updated.
- **SwPreWeld** is executed when the weld position is reached.
- The plate thickness is checked.
- The requested gun force is established.
- The start signal is sent to the weld controller.
- When the weld process is started the program execution continues to the next instruction (if the argument \Corr is used).
- When the weld complete signal from the weld equipment is received, the gun open is started.
- **SwOpenGun** is executed.
- Equalising is deactivated.
- The motion is released and the robot and gun start to move towards the next programmed position.

Motion

The movement to the weld position starts with a synchronous phase which means that the servo gun axis is coordinated with the robot movement.

Gun closure

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

The gun closing will automatically finish synchronised with the robot's approach to the weld position.

If the preclose time is set to 0 in curr_gundata or if the optional argument \InPos is set in the current instruction, then the gun is closed first when the robot has reached the end

position.

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. 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.

The start signal is high during the entire welding period. It is reset either after weld complete or after a predefined timeout time has 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 and other error situations.

Program stop and restart

Stop during the motion and restart

The robot stops on the path. If the asynchronous gun closure is already 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 the next instruction.

Quick stop and restart

Quick stop during the motion and restart

The robot stops immediately and has probably deviated from the path. If the asynchronous gun closure is already started the gun will open to the programmed

position.

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 is reduced.

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.
- The gun is closed and the weld process is 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.

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.

Disable all simulations

All simulations are disabled if *sim_type* = 0 in *curr_simdata*.

Error handling

When *Spot*ML/MJ is stopped by a supervision, the following takes place:

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

The following error situations can occur:

- Instruction parameter supervision.
- Supervision before welding.
- Gun closure supervision.
- Detection of missing or improper plates.
- Weld error.
- Supervision after welding.
- Gun opening supervision.

Instruction parameter supervision

The error occurs when *SpotL/J* 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.

Supervision before welding

The supervisions in *SwPrepare* are executing. This routine is called in the beginning of the motion. See SpotWare Servo - *System Module SWUSER.*

The supervisions can be modified by the user.

Gun closure supervision

Supervisions can be inserted in *SwCloseGun*. This routine is called when the gun starts to close. See Predefined Data and Programs - *System Module SWUSER.*

Detection of missing or improper plates

An error will be detected by the process kernel if the plate thickness differs more than the allowed limit, defined by the tolerance, from the programmed thickness. The error results in an error message on the TP and a program stop.

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.

- The start signal is set low.
- The gun opens.
- The process error signal is set high.
- The following manual choices are available:

  Info / Skip / Reweld (see the dialog box in Figure 1).

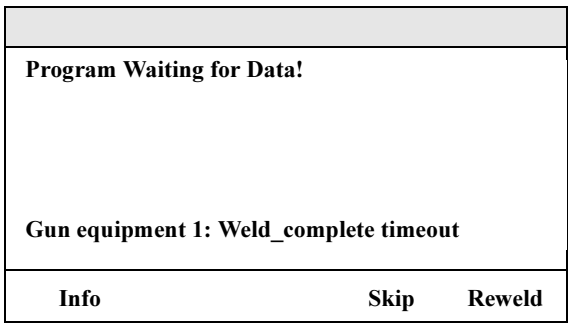


Figure 1 Dialog box for weld error.

Info:

- A dialog box is displayed with more user defined information about the error (if available). See user routine *SwErrorInfo* in Predefined Data and Programs - *System Module SWUSER.*

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.

Supervision after welding

Supervisions in the user defined routine *SwOpenGun* are executing. See Predefined Data and Programs - *System Module SWUSER*. There are no supervisions in the default version of the routine.

To make the execution stop with an error message simply put the desired message in the assigned error string and it will be displayed on the teach pendant.

Gun opening supervision

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

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.

Customising

The SpotWare package gives the user plenty of scope for customising the SpotWare functionality. (See SpotWare Servo - *Customising).*

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

Syntax

SpotL or SpotJ
[ToPoint ‘:=’] < expression (**IN**) of *robtarget* > ‘,’
[Speed ‘:=’] < expression (**IN**) of *speeddata* > ‘,’
[GunNo ’:=’] < expression (**IN**) of *num* > ‘,’
[Spot ’:=’] < persistent (**PERS**) of *spotdata* >
[‘\’ InPos]
[‘\’ Conc]‘,’
[Tool ‘:=’] < persistent (**PERS**) of *tooldata* >
[’\’ WObj ’:=’ < persistent (**PERS**) of *wobjdata* >] ’;’

Related information

	Described in:
Definition of velocity	RAPID Reference Manual - *speeddata*
Definition of zonedata	RAPID Reference Manual - *zonedata*
Definition of tool	RAPID Reference Manual - *tooldata*
Definition of work objects	RAPID Reference Manual - *wobjdata*
Definition of spotdata	SpotWare Servo - *spotdata*
Definition of gundata	SpotWare Servo - *gundata*
SpotML/J	SpotWare Servo - *SpotML/J*
Overview SpotWare Servo	SpotWare Servo - *Summary*
Customising possibilities	SpotWare Servo - *Customising*
I/O configuration	SpotWare Servo - *System parameters*
Servo gun introduction	SpotWare Servo - *Servogun motion control*
Servo gun motion parameters	External Axes
Motion in general	RAPID Reference Manual - *Overview*

1 SpotML/SpotMJ Spot Welding with multiple guns

SpotML and SpotMJ are used in spotwelding if welding with several guns at the same time is desired. For servo guns it is possible to use two guns simultaneously. The instructions are used to control the complete welding sequences, i.e. 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 equipments are activated.

These instructions are only available if the SpotWare Servo Plus option is used.

Example

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

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

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 activate gun equipment 1 and gun equipment 2. *Spotdata* spot10 contains weld parameters for the welding with gun equipment 1, e.g. desired program number and pressure. *Spotdata* spot20 contains weld parameters for the welding with gun equipment 2.

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

Arguments

SpotML ToPoint Speed [\G1] [\G2] [\InPos] [\Conc] Tool [\WObj]

SpotMJ ToPoint Speed [\G1] [\G2] [\InPos] [\Conc] Tool [\WObj]

ToPoint Data type: *robtarget*

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

Speed Data type: *speeddata*

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

[G1] - [G2] Data type: *spotdata*

Spot data with the spot specific data associated with the weld with gun equipment 1 - 2.

[\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.

[\Conc] Data type: *switch*

Normally (without \Conc) the program execution continues to the next instruction when the spot weld process is ready.

With the optional argument \Conc the program execution continues to the next instruction directly after the weld has started and is not blocked until the next order that contains a robot motion. Under some circumstances, this saves cycle time. But be careful, the program pointer is moved to the next instruction and all logical instructions before the next movement instruction are executed during the weld process.

Tool Data type: *tooldata*

The tool in use when the robot moves. The tool centre 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 external axes are used, this argument must be specified in order to perform a linear movement relative to the work object.

Communication

SpotML/MJ communicates with its equipment by using parallel signals.

For a complete description of the I/O configuration, see System Parameters - *Spot Welding*.

Program execution

Internal sequence for each activated gun in a *SpotML/MJ* instruction:

- The robot and guns start to move towards the programmed position.
- **SwPrepare** is executed. (Preweld supervision.)
- The weld program number is set.
- The guns start to close before the position is reached (unless argument \InPos is used), according to the predefined preclosing times.
- **SwCloseGun** is executed.
- Equalising is activated.
- Current spotdata (curr_spotdata in SWUSER) is updated.
- **SwPreWeld** is executed when the weld position is reached.
- The plate thickness is checked.
- The requested gun force is established.
- The start signal is sent to the weld controller.
- When the weld process is started for all activated guns, the program execution continues to the next instruction (if the argument \Corr is used).
- When the weld complete signal from the weld equipment is received, the gun open is started.
- **SwOpenGun** is executed.
- Equalising is deactivated.
- When all activated guns are opened enough the motion is released and the robot and gun start to move toward the next programmed position.

Motion

The movement to the weld position starts with a synchronous phase which means that the servo gun axis is coordinated with the robot movement.

Gun closure

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

The gun closing will automatically finish synchronised with the robot's approach to the weld position.

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

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. 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.

The start signal is high during the entire welding period. It is reset either after weld complete or after a predefined timeout time has 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 and other error situations.

Program stop and restart

Stop during the motion and restart

The robot stops on the path. If the asynchronous gun closure has already started, the guns 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 and has probably deviated from the path. If the asynchronous gun closure is already started the gun will open to the programmed position.

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.

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.
- The guns are closed and the weld process is executed.

Backwards

The motion is performed backwards to the programmed position with gun control.

The guns are not closed in the weld position and no weld processes are activated.

Simulated welding

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

Disable all simulations

All simulations are disabled if *sim_type* = 0 in *curr_simdata*.

Error handling

When *Spot*ML/MJ is stopped by a supervision, the following takes place:

- The signal *process_error* for the interrupted weld process is set.
- An error message is displayed in a dialog box with retry possibilities.
- The error message is logged.

The following error situations can occur:

- Instruction parameter supervision.
- Supervision before welding.
- Gun closure supervision.
- Detection of missing or improper plates
- Weld error.
- Supervision after welding.
- Gun opening supervision.

The supervision activities before and after welding can be modified by the user.

To make the execution stop with an error message simply put the desired message in the assigned error string and it will be displayed on the teach pendant.

Instruction parameter supervision

The error occurs when *SpotML/MJ* 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.

Supervision before welding

The supervisions in *SwPrepare* are executing. See Predefined Data and Programs - *System Module SWUSER*.

Gun closure supervision

Supervisions can be inserted in *SwCloseGun*. This routine is called when the gun starts to close. See Predefined Data and Programs - *System Module SWUSER.*

Detection of missing or improper plates

An error will be detected by the process kernel if the plate thickness differs more than the allowed limit, defined by the tolerance, from the programmed thickness. The error results in an error message on the TP and a program stop.

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.

- The start signal is set low.
- The gun opens.
- The process error signal is set high.
- The following manual choices are available:

  Info / Skip / Reweld (see the dialog box in Figure 1).

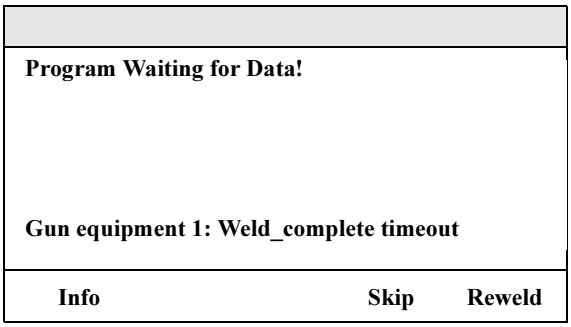


Figure 1 Dialog box for weld error.

Info:

- A dialog box is displayed with more user defined information about the error (if available). See user routine *SwErrorInfo* in Predefined Data and Programs - *System Module SWUSER.*

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.

Supervision after welding

Supervisions in the user defined routine *SwOpenGun* are executing. See Predefined Data and Programs - *System Module SWUSER*. There are no supervisions in the default version of the routine.

To make the execution stop with an error message simply put the desired message in the assigned error string and it is displayed on the teach pendant.

Gun opening supervision

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

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.

Customising

The SpotWare package gives the user plenty of scope for customising the SpotWare functionality. (See SpotWare Servo - *Customising).*

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

Syntax

SpotML or SpotMJ
[ToPoint ':='] < expression (**IN**) of *robtarget* > ','
[Speed ':='] < expression (**IN**) of *speeddata* > ','
['\' G1 ':=' < persistent (**PERS**) of *spotdata* >]
['\' G2 ':=' < persistent (**PERS**) of *spotdata* >]
['\' InPos]
['\' Conc]','
[Tool ':='] < persistent (**PERS**) of *tooldata* >
['\' WObj ':=' < persistent (**PERS**) of *wobjdata* >] ';'

Related information

	Described in:
Definition of velocity	RAPID Reference Manual - *speeddata*
Definition of zonedata	RAPID Reference Manual - *zonedata*
Definition of tool	RAPID Reference Manual - *tooldata*
Definition of work objects	RAPID Reference Manual - *wobjdata*
Definition of spotdata	SpotWare Servo - *spotdata*
Definition of gundata	SpotWare Servo - *gundata*
SpotL/J	SpotWare Servo - *SpotL/J*
Overview SpotWare Servo	SpotWare Servo - *Summary*
Customising possibilities	SpotWare Servo - *Customising*
I/O configuration	SpotWare Servo - *System parameters*
Servo gun introduction	SpotWare Servo - *Servogun motion control*
Servo gun motion parameters	External Axes
Motion in general	RAPID Reference Manual - *Overview*

1 SetForce Close the gun with desired force

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 opens again after the elapsed time or when a digital input signal is set. The instruction can, for example, used for tip dressing.

Example

SetForce gun1, force10;

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

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

Arguments

SetForce GunNo Force

GunNo Data type: *num*

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

Force Data type: *forcedata*

The forcedata with the force parameters.

.

Program execution

Internal sequence when a SetForce instruction is executed:

- The gun is closed
- The plate thickness is checked
- The requested gun force is established
- Wait until the desired force time has elapsed or the force complete signal is activated.
- The gun is opened to the previous position.
- The force data in the array *curr_forcedata* in *SWUSER* is updated.

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

For a complete description of the I/O configuration, see SpotWare Servo - *System Parameters.*

Error handling

Instruction parameter supervision

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

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

Detection of missing or improper plates

An error will be detected by the process kernel if the plate thickness differs more than the allowed limit defined by the tolerance from the programmed thickness. The error results in an error message on the TP and a program stop.

Limitations

Gun equalising is not handled internally when a SetForce instruction is executed.

Syntax

SetForce

[GunNo ':='] < expression (**IN**) of *num* > ','
[Force ':='] < persistent (**PERS**) of *forcedata* > ';'

Related information

	Described in:
Definition of forcedata	SpotWare Servo - *forcedata*
Overview Spot welding	SpotWare Servo - *Summary*
I/O configuration	SpotWare Servo - *System Parameters*

1 CalibL/J Calibrate the gun during movement

CalibL/J is used in spotwelding to calibrate the distance between the gun tips. This is necessary after a tip change or tool change and it is recommended after welding a number of spots or performing a tip dress. Calibrate also updates the tip wear data in the used gundata. The calibration is done during a robot movement to a programmed position. **N.B.** The gun performs two unsynchronised close/open movements during the calibration.

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 equipments used are defined in the *gundata* array *curr_gundata* in SWUSER.

The data curr_tip_wear in curr_gundata is automatically updated.

Arguments

CalibL ToPoint Speed GunNo [\TipChange] | [\ToolChange] | [\TipWear] Zone Tool [\WObj]

CalibJ ToPoint Speed GunNo [\TipChange] | [\ToolChange] | [\TipWear] Zone Tool [\WObj]

ToPoint Data type: *robtarget*

The destination point of the robot and external 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 must not be programmed since this may cause a “Gun calib move error” (error code 50252).

Speed Data type: *speeddata*

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

GunNo Data type: *num*

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

[\TipChange] Data type: *switch*

This calibration type is used after a tip change.

The gun closes and opens two times. The first close movement is slow to find the unknown contact position. The total tip wear is reset to zero.

[\ToolChange] Data type: *switch*

This calibration type is used after a tool change.

The gun closes and opens two times. The first close movement is slow to find the unknown contact position. The total tip wear remains unchanged.

[\TipWear] Data type: *switch*

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 closes and opens quickly two times. The total tip wear is updated.

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 centre 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 external 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 closes and opens 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

Forwards

As during continuous execution.

Backwards

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

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 an error text.

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

Limitations

A movement of the gun tip position must not be programmed since this may cause a “Gun calib move error” (error code 50252).

Syntax

CalibL or CalibJ
[ToPoint ':='] < expression (**IN**) of *robtarget* > ','
[Speed ':='] < expression (**IN**) of *speeddata* > ','
[GunNo ':='] < expression (**IN**) of *num* > ','
[\TipChange] | [\ToolChange] | [\TipWear]','
[Zone ':='] < expression (**IN**) of *zoneddata* > ','
[Tool ':='] < persistent (**PERS**) of *tooldata* >
['\' WObj ':=' < persistent (**PERS**) of *wobjdata* >] ';'

Related information

	Described in:
Overview SpotWare Servo	SpotWare Servo - *Summary*
Servo gun introduction	SpotWare Servo - *Servogun motion control*
Calibration without movement	SpotWare Servo - *Calibrate*

1 Calibrate Calibrate the gun

Calibrate is used in spotwelding to calibrate the distance between the gun tips. This is necessary after a tip change or tool change and it is recommended after welding a number of spots or performing a tip dress. Calibrate will also update the tip wear data in the used gundata. **N.B.** The gun performs two unsynchronised close/open movements during the calibration. The open distance after the calibration is finished will be the same as before the calibration started.

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 equipments 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 [\TipChange] | [\ToolChange] | [\TipWear]

GunNo Data type: *num*

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

[\TipChange] Data type: *switch*

This calibration type is used after a 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.

[\ToolChange] Data type: *switch*

This calibration type is used after a 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*

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 quickly two times. The total tip wear is updated.

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.

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 an error text.

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

Syntax

Calibrate
[GunNo ':='] < expression (**IN**) of *num* > ','
[\TipChange] | [\ToolChange] | [\TipWear]';

Related information

	Described in:
Overview SpotWare Servo	SpotWare Servo - *Summary*
Servo gun introduction	SpotWare Servo - *Servogun motion control*
Calibration with movement	SpotWare Servo - *CalibL/J*

1 TuneGun Tuning Servo Gun

TuneGun is used to tune/change a servo gun parameter. The parameter is changed temporarily from the original value, which is set up in the system parameters. The new tune value will be active immediately after executing the instruction.

TuneGun is useful in tuning procedures. A tuning procedure is typically used to find an optimal value for a parameter. An experiment (i.e. a program execution with a servo gun movement) is repeated when using different parameter tune values.

Example

TuneGun SEOLO_RG, 0.050, CloseTimeAdjust;

The servo gun parameter CloseTimeAdjust is temporarily set to 0.050 seconds.

Arguments

TuneGun MecUnit TuneValue Type

MecUnit Data type: *mecunit*

The name of the mechanical unit.

TuneValue Data type: *num*

New tuning value.

Type Data type: *tunetype*

Parameter type. Servo gun parameters available for tuning are RampTorqRefOpen, RampTorqRefClose, KV, SpeedLimit, CollAlarmTorq, CollContactPos, CollisionSpeed, CloseTimeAdjust, ForceReadyDelayT, PostSyncTime, CalibTime, CalibForceLow, CalibForceHigh. These types are predefined in the system parameters and define the original values.

Description

RampTorqRefOpen

Tunes the system parameter “Ramp when decrease force”, which decides how quickly the force is released when opening the gun. The unit is Nm/s and a typical value 200. The tune may be used during the tuning process of the gun. Note that tune is not allowed when the force is applied.

Corresponding system parameter: Topics *Manipulator*, Type *Force master*,

parameter *ramp_torque_ref_opening.*

RampTorqRefClose

Tunes the system parameter “Ramp when increase force”, which decides how quickly the force is built up when opening the gun. The unit is Nm/s and a typical value is 80. The tune should be used while tuning the gun. Note that tune is not allowed when the force is applied.

Corresponding system parameter: Topics *Manipulator*, Type *Force master*, parameter *ramp_torque_ref_closing.*

KV

Tunes the system parameter “KV”, which is used for speed limitation. The unit is Nms/rad and a typical value is 1. The tune should be used during the tuning process of the gun. For more details, see the external axis documentation. Note that tune is not allowed when force is applied.

Corresponding system parameter: Topics *Manipulator*, Type *Force master*, parameter *Kv.*

SpeedLimit

Tunes the system parameter “Speed limit”, which is used for speed limitation. The unit is rad/s (motor speed) and a typical value is 60. The tune should be used during the tuning process of the gun. Fore more details, see the external axis documentation.Note that tune is not allowed when force is applied.

Corresponding system parameter: Topics *Manipulator*, Type *Force master*, parameter *speed_limit.*

CollAlarmTorq

Tunes the system parameter “Collision alarm torque”, which is used for the automatic calibration of new tips. The tune command should be used while tuning the gun. The unit is Nm (motor torque) and a typical value is 1. Fore more details, see the external axis documentation. Note that tune is not allowed when force is applied.

Corresponding system parameter: Topics *Manipulator*, Type *Force master*, parameter *alarm_torque.*

CollContactPos

Tunes the system parameter “Collision delta pos”, which is used for automatic calibration of new tips. The tune command should be used while tuning the gun. The unit is m and a typical value is 0.002. Fore more details, see the external axis documentation. Note that tune is not allowed when force is applied.

Corresponding system parameter: Topics *Manipulator*, Type *Force master*, parameter *distance_to_contact_position.*

CollisionSpeed

Tunes the system parameter “Collision speed”, which is used for automatic calibration of new tips. The tune command should be used while tuning the gun. The unit is m/s and a typical value is 0.02. Fore more details, see the external axis documentation. Note that tune is not allowed when force is applied.

Corresponding system parameter: Topics *Manipulator*, Type *Force master*, parameter *col_speed.*

CloseTimeAdjust

Constant time adjustment (s), positive or negative, of the moment when the gun tips make contact during a gun closure. May be used to delay the closing slightly when the synchronised pre-closing is used for welding.

Corresponding system parameter: Topics *Manipulator*, Type *SG process*, parameter *min_close_time_adjust*.

ForceReadyDelayT

Constant time delay (s) before sending the weld ready signal after reaching the programmed force.

Corresponding system parameter: Topics *Manipulator*, Type *SG process*, parameter *pre_sync_delay_time.*

PostSyncTime

Release time anticipation (s) of the next robot movement after a weld. This tune type can be tuned to synchronise the gun opening with the next robot movement. The synchronisation may fail if the parameters are set too high.

Corresponding system parameter: Topics *Manipulator*, Type *SG process*, parameter *post_sync_time.*

CalibTime

The wait time (s) during a calibration before the positional gun tip correction is done. For the best results, do not use a too low value, for instance 0.5 s.

Corresponding system parameter: Topics *Manipulator*, Type *SG process*, parameter *calib_time.*

CalibForceLow

The minimum tip force (N) used during a TipWear calibration. For the best results of the thickness detection, it is recommended to use the minimum programmed weld force.

Corresponding system parameter: Topics *Manipulator*, Type *SG process*, parameter *calib_force_low.*

CalibForceHigh

The maximum tip force (N) used during a TipWear calibration. For the best results of the thickness detection, it is recommended to use the max. programmed weld force.

Corresponding system parameter: Topics *Manipulator*, Type *SG process*, parameter *calib_force_high.*

Program execution

The specified tuning type and tuning value are activated for the specified mechanical unit. This value is applicable for all movements until a new value is programmed for the current mechanical unit, or until the tuning types and values are reset using the instruction *TuneGunReset*.

The original tune values may be permanently changed in the system parameters.

The default servo gun tuning values are automatically set:

- by executing instruction *TuneGunReset*
- at a cold start-up
- when a new program is loaded
- when starting program execution from the beginning.

Syntax

TuneGun
[MecUnit ':='] < variable (**VAR**) of *mecunit* > ','
[TuneValue ':='] < expression (**IN**) of *num* > ','
['Type ':='] < expression **(IN)** of *tunetype* >]';'

Related information

	Described in:
Restore of servo gun parameters	Instructions - *TuneGunReset*
Tuning of servo gun	External axes manual

1 TuneGunReset Resetting Servo Gun tuning

TuneGunReset is used to restore original values of servo gun parameters.

Example

TuneGunReset SEOLO_RG;

Restore original values of servo gun parameters for the mechanical unit SEOLO_RG.

Arguments

TuneGunReset MecUnit

MecUnit Data type: *mecunit*

The name of the mechanical unit.

Program execution

The original servo gun parameters are restored

- by executing the instruction *TuneGunReset*
- at a cold start-up
- when a new program is loaded
- when starting program execution from the beginning.

Syntax

TuneGun
[MecUnit ':='] < variable (**VAR**) of *mecunit* > ';'

Related information

	Described in:
Tuning of servo gun	Instructions - *TuneGun*
Tuning of servo gun	External axes manual

spotdata Spot weld data

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

Description

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.
- Allowed variation when checking the plate thickness.

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 - 256 (defined in the system module *SWDEFUSR*).

tip_force *(gun tip force)* Data type: *num*

Defines the desired gun tip force. [N]

plate_thickness *(plate thickness)* Data type: *num*

Defines the expected total plate thickness. [mm]

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.

Predefined data

PERS spotdata spot1 := [1, 1000, 2, 0.5];

Defined in module SWDEFUSR.

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

- The program number 1 in the weld controller shall be used.
- Desired gun tip force = 1000 N.
- Expected total plate thickness = 2 mm.
- Allowed variation in the thickness = 0.5 mm

PERS spotdata curr_spotdata{2} :=

[[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. These parameters are automatically updated by the kernel when spot instructions are executed. This spotdata is used for reweld situations and if welding is manually activated (see Manual Actions in SpotWare Servo - *Programming*).

Customising

The SpotWare package provides opportunities for the user to customise the functionality to adapt to different types of spotwelding 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.

For further details, see SpotWare Servo - *Customising*.

Default Structure

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

Related information

	Described in:
SpotL/J	SpotWare Servo - *SpotL/J*
Overview SpotWare Servo	SpotWare Servo - *Summary*
Customising possibilities	SpotWare Servo - *Customising*
Definition of gundata	SpotWare Servo - *gundata*
System module SWUSER	SpotWare Servo - *System Module SWUSER*

gundata Equipment specific weld data

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.

Description

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

- Gun name
- Closing and equalising times.
- Weld counters and a max. value.
- Current tip wear and a max. value.
- The weld 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 the gun is in the weld position, when the asynchronous gun closure is started.

pre_equ_time *(preequalize time)* Data type: *num*

Time [s] before the gun is in the weld position, when the gun equalising is activated.

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 welds performed. 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.

max_tip_wear *(max tip wear)* Data type: *num*

Max. allowed tip wear before tip exchange [mm]. Use of this data is optional.

weld_timeout *(weld timeout)* Data type: *num*

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

Default Structure

< 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*>

Predefined data

```
PERS gundata curr_gundata{2} :=
                    [["SERVOGUN", 0, 0, 0, 1000, 0, 10, 2],
```

```
["NOT USED", 0, 0, 0, 1000, 0, 10, 2]];
```

Defined in module SWUSER.

curr_gundata is an array with active gundata parameters for each gun used. These parameters must be changed by the user during the installation and programming phase to correspond to the welding equipment in use.

Customising

The SpotWare package provides opportunities for the user to customise the functionality to adapt to different types of spotwelding 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.

For further details, see SpotWare Servo - *Customising*.

.Related information

	Described in:
SpotL/J	SpotWare Servo - *SpotL/J*
Overview SpotWare Servo	SpotWare Servo - *Summary*
Customising possibilities	SpotWare Servo - *Customising*
Definition of spotdata	SpotWare Servo - *spotdata*
System module SWUSER	SpotWare Servo - *System Module SWUSER*

forcedata Spot gun force data

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).

Description

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

- Desired gun tip force.
- Desired force time.
- Expected total plate thickness.
- Allowed variation when checking the plate thickness.

Components

tip_force *(gun tip force)* Data type: *num*

Defines the desired gun tip force. [N]

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]

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.

Predefined data

PERS spotdata force1 := [1000, 1, 0, 0];

Defined in module SWDEFUSR.

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

- Desired gun tip force = 1000 N.
- Desired force time = 1 s.
- Expected total plate thickness = 2 mm.
- Allowed variation in the thickness = 0.5 mm

PERS forcedata curr_forcedata := [[0,0,0,0],[0,0,0,0]];

Defined in module SWUSER.

curr_forcedata is an array with active or latest used forcedata parameters for each defined gun. These 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 in SpotWare Servo - *Programming*).

Customising

The SpotWare package provides opportunities for the user to customise the functionality to adapt to different types of spotwelding 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.

For further details, see Customising SpotWare.

Default Structure

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

Related information

	Described in:
The SetForce instruction	SpotWare - *SetForce*
Overview SpotWare Servo	SpotWare Servo - *Summary*
Customising possibilities	SpotWare Servo - *Customising*
Definition of spotdata	SpotWare Servo - *spotdata*
System module SWUSER	SpotWare Servo - *System Module SWUSER*

simdata Simulation data

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

Description

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.

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)

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.

Predefined data

PERS simdata data curr_simdata := [0, 0.5, FALSE, FALSE];

Defined in module SWUSER.

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

Customising

The SpotWare 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.

For further details, see SpotWare Servo - *Customising*.

Default Structure

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

Related information

	Described in:
The SpotL instruction	SpotWare - *SpotL/J*
The SpotML instruction	SpotWare - *SpotML/J*
Overview SpotWare Servo	SpotWare Servo - *Summary*
Customizing possibilities	SpotWare Servo - *Customizing*
System module SWUSER	SpotWare Servo - *System Module SWUSER*

1 System Module *SWUSER*

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 (e.g. SwPrepare and SwCloseGun). This chapter describes the default functionality.

(See also SpotWare Servo - *Customizing*)

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

- **save SWUSER. The old version is overwritten.**
- **generate a P-Start to affect all tasks.**

Contents

Data

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

The following global data are predefined:

Name	Declaration	Description
curr_gundata{2}	PERS gundata	Current gun specific data for gun equipment 1 and 2.
curr_spotdata{2}	PERS spotdata	Current or latest used spot data for gun equipment 1 and 2. Is automatically updated when weld instructions are running. This data is used when the manual action ManSpot is activated.
curr_forcedata{2}	PERS forcedata	Current or latest used forcedata for gun equipment 1 and 2. Is automatically updated when force instructions are running.
man_forcedata{2}	PERS forcedata	Forcedata for gun equipment 1 and 2 used when manual actions are activated (ManSetForce and ManCloseGun).

Name	Declaration	Description
curr_simdata	PERS simdata	Current parameters for simulation. These parameters have an influence on all equipment in use.

The following signal references are predefined. After the routine SwInitUseIO has been executed they will reference to the physical signal on the activated equipment.

Name	Declaration
reset_fault	VAR signaldo
timer_ready	VAR signaldi
weld_contactor	VAR signaldi
flow1_ok	VAR signaldi
flow2_ok	VAR signaldi
temp_ok	VAR signaldi
air_ok	VAR signaldi
equipment_ok	VAR signaldi

Process Hooks

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 changed. The routines cannot be deleted since they are called from internal modules.

Parameters description for the process hooks:
GunNum: Gun equipment number

ErrText: Error message. An error text in this parameter will generate an error dialog with possibilities for the operator to decide what to do. If ErrText = TxtRetry 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 and is 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 signals used. As default, only signals for two gun equipments 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.

No default functionality.

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 some status signals from the weld timer are 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:

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 pressed in an error dialog. It is possible to add extra error information here.

Default functionality: The text strings ErrType and ErrText are written on the display plus the text string “No further information”. A push button “Return” ???

Possible ErrType values:

TxtSwPreWeld	Error generated in SwPreWeld.
TxtSwCloseGun	Error generated in SwCloseGun.
TxtSwPrepare	Error generated in SwPrepare
TxtSwOpenGun	Error generated in SwOpenGun
TxtGenWeldTitle	Weld error

PROC SwResetFault(num GunNum)

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

Default functionality: The reset signal for the current gun is pulsed.

1 System Parameters

The digital and analog signals used for SpotWare are configured in the system parameters.

• Choose **Topics: IO Signals**

• Choose **Types: User Signals**

See User´s Guide - *System Parameters*.

1.1 IO configuration

The SpotWare package can be configured for different equipment setups. This chapter describes the basic (default) setup used by SpotWare Servo and SpotWare Servo Plus.

The basic setup is for two gun equipments but it is possible to configure up to eight gun equipments in a similar way. This signal configuration is sufficient for running the default version of the SpotWare Servo or SpotWare Servo Plus package.

The physical connections can be changed freely. To save physical signals, signals not in use can be connected to a simulated (SIM) board (type eip000).

1.2 Basic setup - Board description

There are three predefined boards:

- One physical digital I/O board, named SW_BOARD1, on address node 10. This board are used for basic setup signals for gun equipment 1.
- One virtual board, named SW_BOARD2 with basic setup signals for gun equipment 2.
- One virtual board, named SW_SIM_BOARD with some internal or normally not connected signals

1.3 Basic setup - Signal description

Weld timer signals for gun 1

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
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 SpotWare Servo - *Customising*.
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 SpotWare Servo - *Programming*.
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 timeout.
g1_timer_ready	input	The timer is ready to weld.

N.B. Signals in the same group must be connected to physical signals in sequence on the same board.

Gun signals for gun 1

g1_equalize	output	Gun equalize signal
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 are cross-connected to this signal.

Process status signals for gun 1

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.

Signal names for gun 2 are the same as for gun 1 but prefixed with g2_

Other signals

force_complete	input	Can be used to interrupt the SetForce instruction before the programmed force time elapsed.
reweld_proc	input	Can be used to answer a weld error dialog with an input signal. The same as pressing “REWELD”.
skip_proc	input	Can be used to answer an error dialog with an input signal. The same as pressing “SKIP”.

1 Customising

The SpotWareServo package provides wide opportunities to customise the functionality to adapt to different types of spotwelding equipment and user standards. Another purpose of this customising process is to reduce the amount of data and number of variables presented to the programmer or operator.

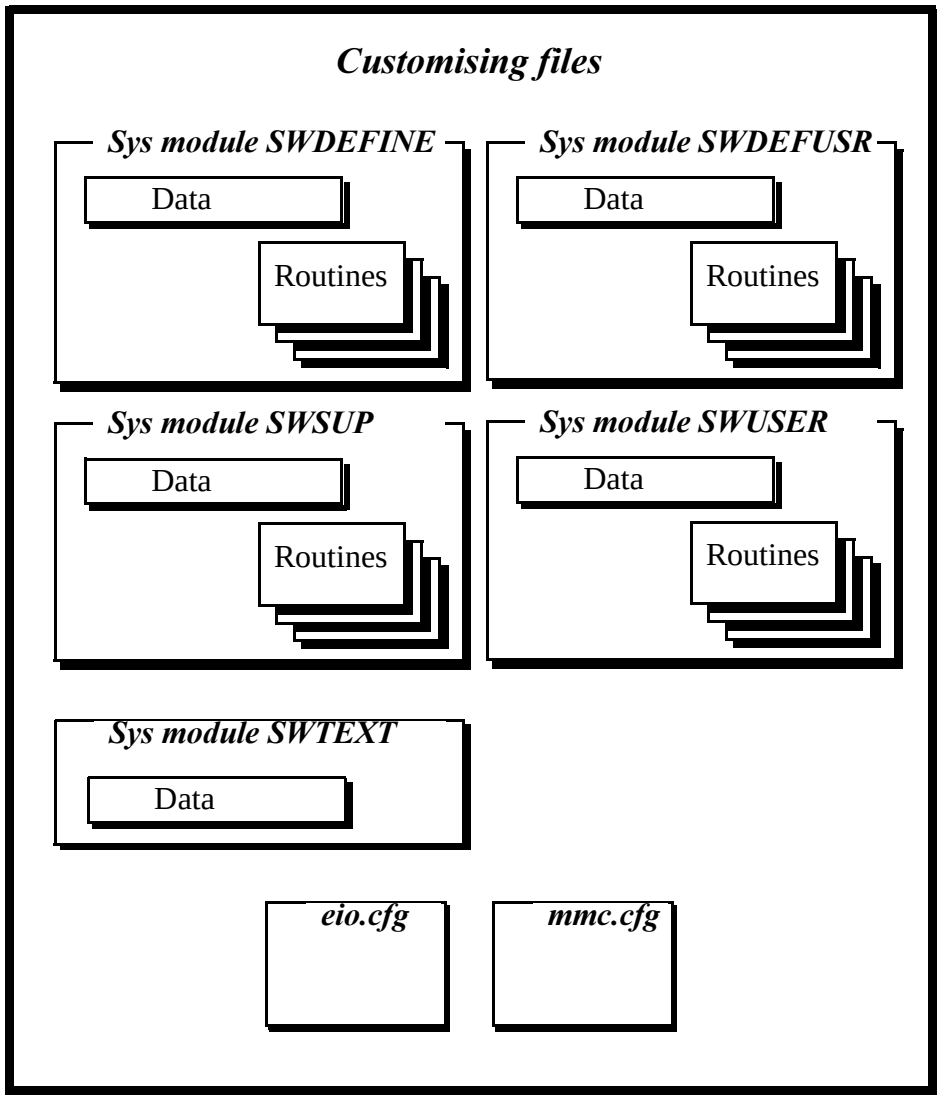
1.1 Customising possibilities

Examples:

- Define the number of guns to be used. Only used guns are visible and programmable on the teach pendant.
- 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 SpotWare data types to the spot weld equipment in use.
- Adapt the names of the data components in the SpotWare 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 (or delete) 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 menu Special in the Program Window.
- Create other equipment specific Programming instructions (e.g. 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.
- Decide which routines and data will be visible to the programmer.
- Define the MostCommon I/O list with your most frequently used signals.
- Define the MostCommon instruction pick list.
- Change the predefined text strings.
- Use data from the weld timer and not from spotdata (e.g. tip force).

1.2 Files intended to be changed during the customising process

The customising process is done by changing a number of predefined data and routines, preferably using a standard PC. The following RAPID modules and configuration files are intended to be changed during the customising process:



Figur 1 Files intended for customising.

Short file descriptions:

SWDEFINE (Readonly)

This module is running in the Main task and contains *event routines* and routines used for *manual actions*.

SWDEFUSR (Noview)

This module is running in the Main task and all process tasks and contains data and routines for the definition of the different SpotWare data types. It also includes a number of setup data and definitions of max. and min. values for some SpotWare data components.

SWUSER

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 teach pendant.

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 (e.g. SwPrepare and SwCloseGun). It is possible to move data and routines from SWUSER to SWDEFUSR and vice versa, depending on whether they shall be possible to change from the teach pendant or not.

(See also SpotWare Servo - *System Module SWUSER*)

SWTEXT (BuiltIn, Shared)

Module containing text strings for the SpotWare Servo option.

SWSUP (Noview)

Module containing the routines for the autonomous supervision and signal control. SWSUP is running as a separate semi-static task.

In the default version, for example, the weld_power signals are activated and the water_start signal is pulsed at MotorsOn.

I/O configuration (eio.cfg)

Default spot weld signals for two gun equipments are defined but all signals are connected to a simulated board. The signals used must be connected to physical signals. Signals not used must remain on the simulated board (see SpotWare Servo - System Parameters).

MMC configuration (mmc.cfg)

This configuration file contains 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.

1.3 How to change the number of guns to be used

As default it is possible to use two guns but it is possible to use and set up data for up to eight different guns in the system.

- Change the data SW_NOF_GUNS in SWDEFUSR to the desired value (1-8).
- Add signals in eio.cfg for all equipments to be used in a similar manner as the predefined signals for gun equipment 1 and 2.
- Increase the number of data in the following arrays in SWUSER (The size must comply with the number of guns used):
  - curr_gundata
  - curr_spotdata
  - curr_forcedata
  - man_forcedata
- Increase the routines SwInitUserIO in SWUSER and DefSupIO in SWSUP with AliasIO instructions for the signals corresponding to the extra gun equipments.

1.4 How to define max. and min. values for a number of data components

It is possible to define the max. and min. values for a number of data components. The limits will be tested at runtime.

• Change the following data in SWDEFUSR to the desired value.

Data	Description	Default value
MAX_TIP_FORCE	Max value for tip_force in spotdata and forcedata [N]	6000
MAX_PLATE_THICK	Max value for plate_thickness in spotdata and forcedata [mm]	40
MAX_PLATE_TOLER	Max value for plate_tolerance in spotdata and forcedata [mm]	1
MAX_PROG_NO	Max value for prog_no in spotdata	64
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]	1
MAX_FORCE_T	Max value for force_time in forcedata [s]	10

1.5 How to change the SpotWare data types

• Change the definition of the SpotWare data types in SWDEFUSR to the 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.

• Change the corresponding instructions in the data definition routines in SWDEFUSR. These routines are used to connect the user defined data components to internal data.

• 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

• Change text strings in SWTEXT if necessary.

1.6 How to add functionality in the process sequence

• Add code to the process hooks in SWUSER

- See description of the process hooks in SpotWare Servo - *System Module SWUSER*

1.7 How to change, add or delete Manual Actions

• Change the routines for the Manual Actions. This routines (ManSpot, ManCloseGun, ManOpenGun etc.) are found in the module SWDEFINE.

• If Manual Actions are added or deleted, or if Manual Action names are changed, then the MMC configuration must also be changed. Change the names under MMC_SERV_ROUT_STRUCT:

1.8 How to create other equipment specific programming instructions

• Create global routines with the desired name, syntax and functionality. Use a system module with the attribute NOVIEW or NOSTEPIN.

• The MMC configuration must be changed to define the instruction syntax and to get the instruction into an instruction pick list.

1.9 How to add equipment specific supervision and error handling

• 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 SpotWare Servo - *System Module SWUSER*

- If the autonomous supervision must be changed, the code in the supervision module SWSUP must 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.

1.10 How to use own signal names on internal used signals

- The following routine calls in SWDEFINE are used to inform the kernel about current signal names:
  - DefGunIO
  - DefInternalIO
  - DefTimerIO
- Change the parameters in these instructions so they comply with the used signal names.

1.11 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 (e.g. on separate input groups). Then it is possible to use this data instead of the corresponding data from current spotdata.

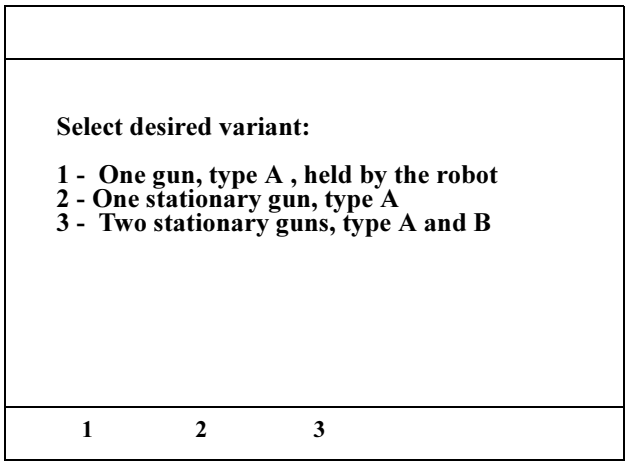
- Add code in SwCloseGun in SWUSER to transfer this data to the internal data, e.g. int_spotdata{GunNum}.tip_force := g1_tip_force.
- Remove data components from spotdata that are not used, see **How to change the SpotWare data types**.

1.12 How to package and install the result from the customising process

After customising, it is a good idea to create a new map with the changed files for each customised variant and load it as an external option. During the start up sequence the user is then given an opportunity to pick the desired variant from a list on the teach pendant.

When the external option is loaded the default system modules are replaced by the customised modules and the influenced cfg files are replaced of the customised versions for the desired variant.

See example in the following figure:



Figur 2 Equipment type selection example.