<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Programming SpotWare</strong></td>
<td>3</td>
</tr>
<tr>
<td>Programming Spot Welding</td>
<td>3</td>
</tr>
<tr>
<td>Tip-dressing</td>
<td>5</td>
</tr>
<tr>
<td>Defining gun closing times</td>
<td>6</td>
</tr>
<tr>
<td>Customizing the spot weld instruction</td>
<td>7</td>
</tr>
<tr>
<td><strong>Spot Welding</strong></td>
<td>9</td>
</tr>
<tr>
<td>IO configuration</td>
<td>9</td>
</tr>
<tr>
<td>Basic setup</td>
<td>9</td>
</tr>
<tr>
<td>Double gun</td>
<td>12</td>
</tr>
<tr>
<td>Equipment control</td>
<td>12</td>
</tr>
<tr>
<td>Manual Actions</td>
<td>12</td>
</tr>
<tr>
<td>Process state</td>
<td>13</td>
</tr>
<tr>
<td><strong>gundata - Spot weld gun data</strong></td>
<td>15</td>
</tr>
<tr>
<td>Description</td>
<td>15</td>
</tr>
<tr>
<td>Components</td>
<td>15</td>
</tr>
<tr>
<td>Structure</td>
<td>17</td>
</tr>
<tr>
<td>Related information</td>
<td>17</td>
</tr>
<tr>
<td><strong>spotdata - Spot weld data</strong></td>
<td>19</td>
</tr>
<tr>
<td>Description</td>
<td>19</td>
</tr>
<tr>
<td>Components</td>
<td>19</td>
</tr>
<tr>
<td>Example</td>
<td>20</td>
</tr>
<tr>
<td>Structure</td>
<td>20</td>
</tr>
<tr>
<td>Related information</td>
<td>20</td>
</tr>
<tr>
<td><strong>SpotL/SpotJ - Spot Welding with motion</strong></td>
<td>21</td>
</tr>
<tr>
<td>Example</td>
<td>21</td>
</tr>
<tr>
<td>Arguments</td>
<td>21</td>
</tr>
<tr>
<td>Customizing</td>
<td>22</td>
</tr>
<tr>
<td>Program execution, overview</td>
<td>23</td>
</tr>
<tr>
<td>Program execution, detailed</td>
<td>24</td>
</tr>
<tr>
<td>Program stop and restart</td>
<td>27</td>
</tr>
<tr>
<td>Instruction by instruction execution</td>
<td>27</td>
</tr>
<tr>
<td>Simulated welding</td>
<td>28</td>
</tr>
<tr>
<td>Error handling</td>
<td>28</td>
</tr>
<tr>
<td>Manual actions</td>
<td>31</td>
</tr>
<tr>
<td>Communication</td>
<td>32</td>
</tr>
<tr>
<td>Power failure handling</td>
<td>32</td>
</tr>
<tr>
<td>Syntax</td>
<td>32</td>
</tr>
</tbody>
</table>
Programming SpotWare

The SpotWare package is added to the base system and is ready to use. However, in order to be able to adapt the package to special needs, the I/O configuration and application specific program data and routines can be modified. See Customizing the spot weld instruction on page 7.

Programming Spot Welding

Spot weld instructions

A spot weld instruction comprises both motion and process. It contains basically the same types of information as a positioning instruction. However, each spot weld instruction includes further arguments, spot and gun, that serve as data for the spot weld process (data types: spotdata and gundata).

<table>
<thead>
<tr>
<th>SpotL/J</th>
<th>p1, v100, spot1 \Inpos \NoConc \Retract, gun1, tool1 \frame1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data for the spot weld process</td>
<td>Switch inhibiting gun pre-closing</td>
</tr>
<tr>
<td>Switch inhibiting concurrent execution</td>
<td>Choice for opening gap after the weld</td>
</tr>
<tr>
<td>Gun specific spot weld data</td>
<td>Arguments as in the MoveL/MoveJ instruction</td>
</tr>
</tbody>
</table>

Each SpotL/J instruction includes a movement to the weld position and one weld.

Since the spot weld is always carried out at a stop point the zone argument is omitted.

Note. gun1 and tool1 contain data for the same tool, i.e. the spot weld gun, where tool1 contains the motion-specific data known from the MoveL/J instruction.

Defining spot weld data

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

- spotdata; describes the spot weld process-specific data
- gundata; describes spot weld gun-specific data

There is already data defined in the spot weld system module SWURSF.
Manual gun control

The gun control is by default done by digital outputs.

To ensure fast access to the gun, design your own I/O preference list with the gun action controls on top.

`SwOpenGun / SwCloseGun` are signals which initiate actions. Upon activation they execute the user defined routines `sw_close_gun / sw_open_gun`. These routines use by default the signals `close_tip1 / close_tip2` for opening / closing and the signal `work_select` for choosing the opening gap (work = small = 1 / retract = big = 0). The routines also use the internal gundata of the last SpotL instruction. To make changes in the internal gundata first put the changes in `man_gun` and then set the signal `SwNewData`. `man_gun` is located in the system module SWUSRC. To change the opening gap use `man_retr` in the system module SWUSRC.

`SwOpenGun / SwCloseGun` for closing and opening the gun.

`SwNewData` for transferring the contents of `man_gun` to the internal gundata and `man_retr` to set the opening gap.

Manual weld

A complete weld, including supervision and error recovery, may be run independently of the positioning. It is activated by the signal `SwRunProc` triggering the attached action. The same principle as for manual gun manipulation is used to make changes in the internal spotdata. Put the changes in `man_spot` located in SWUSRC.

`SwRunProc` for executing a spot weld including gun open/close.

`SwNewData` for transferring the contents of `man_spot` to the internal spotdata

Programming spot weld instructions

- Jog the robot to the desired destination 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 spot weld instruction. (See Figure 1).
Figure 1  A spot weld instruction is added directly to the program.

- Change the arguments if necessary.

Testing spot weld instructions

To prevent the spot weld process executing during programming, it is possible to run SpotL/J in simulation mode.

This can be done by setting \textit{sw\_inhib\_weld} TRUE. This will set the output signal \textit{current\_enable} low and the simulation will be carried out by the weld timer.

If such a signal is not connected the spot weld can be internally inhibited by setting \textit{sw\_sim\_weld} TRUE. The simulated weld time is then stored in \textit{sw\_sim\_time}. In this simulation mode the start signal is never sent to the welding timer.

\textbf{Note.} The gun will be opened and closed as normal.

Running spot weld instructions

In continuous mode the spot weld instructions are by default running in concurrent execution, i.e. the program execution is continuing while welding is in progress, and it will usually stop at next motion instruction.

Therefore if the running is stopped during welding, the program pointer is normally located on the next SpotL/J or motion instruction. It is important to remember this fact when modifying SpotL/J positions with \textit{ModPos}.

To avoid problems when positions are modified it is recommended to execute the program in step-by-step mode. In this mode the program pointer corresponds to the robot position. If a strictly sequential execution is desired also in continuous mode the \textit{NoConc} argument must be set in the instruction.

Tip-dressing

The gundata contains counters and limits for each pair of tips. The counters will be
incremented for each spot.

A tip-dress supervision can however be programmed in the main program or in the SpotL supervision routines.

In this way, an automatic tip-dressing can be performed.

---

**Defining gun closing times**

The SpotL/J instruction has a built-in pre-closing of the weld gun, i.e. when approaching the position the gun will start to close in advance, in order to save time. The pre-closing relates to closing the gun from the work stroke. Closing the retract stroke should be handled in the user program, by a fly-by position.

The gun closing times are defined in gundata. It is possible to define a closing time for each gun pressure (max. 4).

The pressure to use in the actual SpotL/J instruction is stored in spotdata. It will also pick the accurate closing time.

**Note.** The pre-closing can be disabled by choosing the \InPos argument in the instruction or by setting \textit{sw\_inpos} TRUE in system module SWURSF (global disabling).

**Defining the closing time manually**

- First, define a rough guess in the actual gundata
- Define the pressure in spotdata
- Open the test window
- Choose execution mode \textit{Instruction}
- Run SpotL/J forward and check the closing. The gun should be just about closed when the robot stops in the position.
- If the result is not OK, execute SpotL/J backwards. The gun will open automatically before it moves.
- Adjust the corresponding closing time.
- Repeat the test until the result is OK.
- Repeat the whole sequence for a different pressure and different stroke until all closing times are defined.

**Defining the closing time automatically**

It is possible to measure and store the closing times using a RAPID program.
Customizing the spot weld instruction

Customizing can be done at different levels:

- SWUSRF, SWUSRC: Modules containing routines and data used by SpotL/J
- I/O configuration.

**SWUSRF**

Global data, service and equipment routines affecting the entire program.  
(See Predefined Data and Programs). This module will be called from task 0.

**SWUSRC**

Global data, service and equipment routines affecting the entire program.  
(See Predefined Data and Programs). This module will be called from task 1 and task 2. To have the possibility to change the content in the module it’s also loaded in task 0.

**I/O configuration**

Mirroring the connected equipment.  
(See System Parameters).
Figure 2  The SpotWare environment.
System Parameters SpotWare

The spot welding system parameters are defined by input and output signals.

- Choose **Topics: IO Signals**
- Choose **Types: User Signals**

See Chapter 12, System Parameters.

IO configuration

The SpotWare package can be configured for different equipment setups. This chapter describes the signals used by SpotWare and their dependency of the equipment.

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

If different signal names are required, new logical signals can be added and connected to the corresponding physical signal. The predefined logical name must remain in the configuration as it is used internally by the SpotL instruction.

Basic setup

There are two predefined boards:

- one physical board named SW_BOARD with default setup:
  - digital I/O board
  - address 10
- one virtual, named SIM_BOARD.

The basic setup signals are connected to the SW_BOARD. All others are connected to the SIM_BOARD which makes them easy to activate.
Start and monitoring of the weld

\[\text{start1} \quad \text{output 7}\quad \text{Start signal to the weld timer (tip1 if a dual gun is used)}\]

\[\text{weld\_ready1} \quad \text{input 7}\quad \text{A high signal is obtained from the welding timer when the weld, started with start1, is finished. The signal is expected to be reset by the weld-timer when the start signal is reset. If the weld\_ready signal is not received within a pre-defined time period (sw\_wr\_timeout), an error message will be displayed.}\]

Weld program number

\[\text{prog\_no\_group} \quad \text{output 11-16}\quad \text{Weld program number}\]

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

Gun control

\[\text{close\_tip1} \quad \text{output 5}\quad \text{Close gun signal. (Tip 1, if a dual gun is used)}\]

\[\text{work\_select} \quad \text{output 6}\quad \text{Work stroke select. The signal is set after the welding according to the Retract argument in the instruction. If work stroke (small gap) is desired the signal is set to 1. If retract stroke (large gap) is desired the signal is set to 0.}\]

Pressure setting and supervision

\[\text{pressure\_group} \quad \text{output 2-4}\quad \begin{align*} p2\_req \quad p4\_req \end{align*}\]

The SpotWare can set and supervise up to four discrete pressure levels. The pressure value are selected in spotdata. The number of possible pressure levels is defined for each gun by nof\_plevels in gundata. The corresponding output signals p2\_req to p4\_req must be defined as consecutive physical outputs. When only one pressure level exists, no pressure request signals need to be connected.

<table>
<thead>
<tr>
<th>nof_plevels</th>
<th>output</th>
<th>output group</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>none</td>
<td>pressure group (simulated outputs)</td>
</tr>
<tr>
<td>2</td>
<td>p2_req</td>
<td>pressure group (p2_req)</td>
</tr>
<tr>
<td>3</td>
<td>p3_req</td>
<td>pressure group (p2_req, p3_req)</td>
</tr>
</tbody>
</table>
If supervision of the ordered pressure value is required before welding \((\text{close\_request} = \text{TRUE in gundata})\), the supervision is done on one of the input signals \(p1\_\text{ok}\) to \(p4\_\text{ok}\) depending on the selected pressure value in spotdata.

Gun opening supervision

If \(\text{open\_request} = \text{TRUE in gundata}\), the input signal \(\text{tip1\_open}\) will be checked whether or not the gun is open before motion is released. If a dual gun is used the input signal \(\text{tip2\_open}\) is also checked.

Timer reset

\[
\text{reset\_fault} \quad \text{output 1}
\]
Reset signal. Can be used to reset the welding controller after a weld error. The signal is pulsed with a user defined pulse length \((\text{sw\_reset\_time1})\) before manual or automatic rewelding.

Process error

\[
\text{process\_error} \quad \text{output 9}
\]
Operator request is set when an error situation occurs.

Current signal

\[
\text{current\_enable} \quad \text{output 8}
\]
Current enable signal. Used for the weld inhibit function. The signal is set to 1 if \(\text{sw\_inhib\_weld} = \text{FALSE}\).

Equipment supervision

By default, the following input signals are tested in every welding position before the weld start. If desired, other user defined signals can be tested, before or after the welding process.

\[
\text{timer\_ready} \quad \text{input 8}
\]
The timer is ready to weld.

\[
\text{flow\_ok} \quad \text{input 9}
\]
Signal indicating problems with the water supply.

\[
\text{temp\_ok} \quad \text{input 10}
\]
Signal indicating over-temperature.
Double gun

If a double gun is used it has to be indicated in gundata by nof_tips = 2. The data tip_no in spotdata chooses the tip (tip_no = 12 or 21 closes both tips together).

- start2 sim output
  Start signal, tip2, to the weld timer.

- weld_ready2 sim input
  A high signal is obtained from the welding timer when the weld, started with start2, is finished. The signal is expected to be reset by the weldtimer when the start signal is reset. If the weld ready signal is not received within a predefined time period (sw_wr_timeout), an error message will be displayed.

- close_tip2 sim output
  Close gun request, tip 2

Equipment control

By default, the following input signals are set depending on the motor on state (motor_on) and the output signals current_enable and process_error.

- weld_power sim output
  Possible to use for a weld power unit contactor

- water_start sim output
  Possible to use to activate water cooling

<table>
<thead>
<tr>
<th>Activators</th>
<th>motor_on</th>
<th>process_error</th>
<th>current_enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 1 0 1 0</td>
<td>0 1 0 1</td>
<td>0 1 0 1</td>
</tr>
<tr>
<td></td>
<td>0 0 1 1 0</td>
<td>0 0 1 1</td>
<td>0 0 1 1</td>
</tr>
<tr>
<td></td>
<td>0 0 0 0</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Result</th>
<th>water_start</th>
<th>weld_power</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 0 0 0</td>
<td>0 0 pulse 0 0</td>
</tr>
<tr>
<td></td>
<td>0 0 0 0</td>
<td>0 0 1 0 0</td>
</tr>
</tbody>
</table>

Manual Actions

The manual action signals are internally triggering RAPID routines to execute. The signals are defined on the SIM_BOARD.
The manual actions operate on internal data. This data may be updated in three ways:

- Running SpotL - the internal data contains the contents of the SpotL/J-parameters.
- Set the output SwNew Data - the internal data contains the contents of the manual data `man_spot`, `man_gun` and `man_retr`.
- Using the utility routines SwSetCurrSpot, etc. (See System module SWTOOL).

SwNewData sim output Transfers manual data to internal data

SwCloseGun sim output Close the gun, runs `sw_close_gun`

SwOpenGun sim output Open the gun, runs `sw_open gun`

SwRunProc sim output Runs a complete spot weld including opening/closing of the gun independently of the motion. This action may also be used to run a reweld when the process has stopped in a weld ready timeout situation.

SwSkipProc sim output Skip the ongoing SpotL/J-instruction and be ready for the next instruction. As long a weld is not ongoing first the user defined routine `sw_open_gun` will be called and then the system will cleanup the ongoing SpotL/J-instruction. This signal is also used by SpotWare for internal process abortion.

Process state

The following signals used internally and give the information about the state of the SpotWare process:

proc_run sim output The signal is set to 1 at motion start and set to 0 again at the end of the ongoing SpotL/J-instruction, i.e. after cleanup.

inhib_move sim output The signal is set to 1 at motion start. If the switch `\NoCon` is used the signal is set to 0 again after cleanup otherwise the signal is set to 0 just after the start signal is sent to the weld timer. When the signal is 1 PP is blocked within the ongoing SpotL/J-instruction.

weld_error sim output Set when a weld ready timeout has occurred. Reset when re-welding was successful or after skipping the current weld.
## REVISION PAGE

**Document Registration number:** SWD94007.ARD

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hemrik Knobel</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>KCF</strong></td>
<td><strong>foxsw</strong></td>
<td></td>
</tr>
<tr>
<td><strong>KCF</strong></td>
<td><strong>swd</strong></td>
<td></td>
</tr>
<tr>
<td><strong>KCF</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MT</strong></td>
<td><strong>Mats Källman</strong></td>
<td></td>
</tr>
</tbody>
</table>

: User Manual | MP |
: User Manual | MT |
Fact: User Manual | KC |
: User Manual | MT |

## REVISION

<table>
<thead>
<tr>
<th>Revision</th>
<th>Scope</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>All</td>
<td>First release</td>
<td>940812 KCF/HK</td>
</tr>
<tr>
<td>0.11</td>
<td></td>
<td>Rw 2.0 status</td>
<td>950503 KCF/ÅO</td>
</tr>
<tr>
<td>0.12</td>
<td></td>
<td>Weld ready signal missing in text</td>
<td>981015 TS</td>
</tr>
<tr>
<td>0.13</td>
<td></td>
<td>More detailed description</td>
<td>991125 HE</td>
</tr>
</tbody>
</table>
**gundata**  

**Gundata** is used to define spot weld gun specific data, to control the gun in an optimal way in the weld process.

**Description**

Gundata is used in spot weld instructions and has the following structure:

- Number of electrode pairs.
- Number of pressure levels.
- Flag to indicate if a signal is required to test accurate gun closure.
- Flag to indicate if a signal is required to test accurate gun opening.
- Counter for the number of welds done and maximum allowed number (one counter and maximum value for each pair).
- Closing times.
- Opening times.

**Components**

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>nof_tips</td>
<td>Number of electrode pairs (1 or 2).</td>
<td>num</td>
</tr>
<tr>
<td>nof_plevels</td>
<td>Number of pressure levels (1 - 4).</td>
<td>num</td>
</tr>
<tr>
<td>close_request</td>
<td>If the flag is TRUE, the ordered pressure level is tested before the weld may start.</td>
<td>bool</td>
</tr>
<tr>
<td>open_request</td>
<td>If the flag is TRUE, the gun open signal is tested before the next motion is released. The time out is defined by <em>sw_go_timeout</em>.</td>
<td>bool</td>
</tr>
</tbody>
</table>

**Note.** The *open_time* must elapse before the test is done (see below).

If the flag is FALSE, the next motion is always released after *open_time* (see below).

<table>
<thead>
<tr>
<th>Component</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>tip1_counter</td>
<td>num</td>
</tr>
</tbody>
</table>

Counter for the number of welds done with the first pair of electrode tips. The counter is automatically incremented. Use of counter is optional. The zeroing shall be handled by the user program.
**gundata** Spot weld gun data

**SpotWare**

tip2_counter

Data type: *num*

Counter for the number of welds done with the second pair of electrode tips. The counter is automatically incremented. Use of counter is optional. The zeroing shall be handled by the user program.

tip1_max

1

Data type: *num*

Maximum number of welds to be made by the first pair of electrode tips before tip service is required. This parameter may be used for automatic tip dressing etc.

tip2_max

1

Data type: *num*

Maximum value for number of welds allowed with the second pair of electrode tips before tip service shall be performed. To be used if desired by the user program.

close_time1

Data type: *num*

Time [s] to close the gun if pressure p1 is activated.

close_time2

Data type: *num*

Time [s] to close the gun if pressure p2 is activated.

close_time3

Data type: *num*

Time [s] to close the gun if pressure p3 is activated.

close_time4

Data type: *num*

Time [s] to close the gun if pressure p4 is activated.

build_up_p1

1

Data type: *num*

Time [s] elapsed to build up the pressure from the moment when the gun has just about closed. The pressure p1 has to be set in advance. This parameter may be used to measure the closing time automatically.

build_up_p2

1

Data type: *num*

Time [s] elapsed to build up the pressure from the moment when the gun has just about closed. The pressure p2 has to be set in advance.

build_up_p3

1

Data type: *num*

Time [s] elapsed to build up the pressure from the moment when the gun has just about closed. The pressure p3 has to be set in advance.

build_up_p4

1

Data type: *num*

Time [s] elapsed to build up the pressure from the moment when the gun has just

---

1. To be defined only if used by the user program.
about closed. The pressure p4 has to be set in advance.

**open_time**  
Data type: *num*

The time [s] that always elapses between the gun opening order and the release of the next motion or the test of gun open (see above).

### Structure

```xml
< dataobject of gundata>
  <nof_tips of num>
  <nof_plevels of num>
  <close_request of bool>
  <open_request of bool>
  <tip1_counter of num>
  <tip2_counter of num>
  <tip1_max of num>
  <tip2_max of num>
  <close_time1 of num>
  <close_time2 of num>
  <close_time3 of num>
  <close_time4 of num>
  <build_up_p1 of num>
  <build_up_p2 of num>
  <build_up_p3 of num>
  <build_up_p4 of num>
  <open_time of num>
</ dataobject of gundata>
```

### Related information

<table>
<thead>
<tr>
<th>Spot weld instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition of tool TCP, weight etc.</td>
<td>Instructions - <em>SpotL</em></td>
</tr>
<tr>
<td>Data types - <em>tooldata</em></td>
<td></td>
</tr>
</tbody>
</table>

---

*SpotWare*
SpotData is used to define the parameters that control a weld timer and weld gun for welding a certain spot.

Description

Spotdata is referred to spot weld instructions and contains data which controls the welding in the actual instruction.

Spotdata has the following structure:

- Program number for the program in the weld timer to be used.
- Electrode tip to be activated (in case of a double gun).
- Desired gun pressure (if connected).
- Weld timer number (if more than one).

Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prog_no</td>
<td>(program number) Data type: num</td>
</tr>
<tr>
<td></td>
<td>Defines the internal program in the weld timer to be used for the welding. The allowed range is 0..&lt;sw_prog_max (defined in the system module SWUSRC).</td>
</tr>
<tr>
<td>tip_no</td>
<td>(electrode pair number) Data type: num</td>
</tr>
<tr>
<td></td>
<td>Defines the electrode pair to be activated. The following alternatives are available:</td>
</tr>
<tr>
<td></td>
<td>1: Electrode pair 1</td>
</tr>
<tr>
<td></td>
<td>2: Electrode pair 2</td>
</tr>
<tr>
<td></td>
<td>12: Electrode pairs 1 and 2. Both pairs are closing at the same time. The welding is done in sequence, pair 1 first and pair 2 second. Then, both electrodes are opening together.</td>
</tr>
<tr>
<td></td>
<td>21: Electrode pairs 2 and 1. Both pairs are closing at the same time. The welding is done in sequence, pair 2 first and pair 1 second. Then, both electrodes are opening together.</td>
</tr>
</tbody>
</table>

For guns with only one pair of electrodes the value shall be 1.
**spotdata Spot weld data**

**gun_pressure**  
*(gun pressure)*  
Data type: *num*

Defines the gun pressure to use.

The following alternatives are available: Gun pressure 1 - 4

**timer_no**  
*(weld timer number)*  
Data type: *num*

Defines the weld timer to be used. Only used for certain type of weld timers.

If only one timer is used (normal case) the value shall be 1.

---

**Example**

PERS spotdata spot1:= [16, 1,4,1];

The spotdata spot1 is programmed for an equipment containing one single gun and one weld timer. When spot1 is used the following occurs:

- The program number 16 is controlling the welding.
- The gun pressure must reach level 4 before the weld is allowed to start.

---

**Structure**

<dataobject of spotdata>

<prog_no of num>

<tip_no of num>

<gun_pressure of num>

<timer_no of num>

---

**Related information**

<table>
<thead>
<tr>
<th>The Spot weld instruction</th>
<th>Described in:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gundata</td>
<td>Instructions - SpotL</td>
</tr>
<tr>
<td></td>
<td>Data Types - Gundata</td>
</tr>
</tbody>
</table>
SpotL/SpotJ Spot Welding with motion

SpotL (SpotLinear) and SpotJ (SpotJoint) is used in spotwelding to control the motion, gun closure/opening and the welding process. SpotL moves the TCP linearly to the end position. SpotJ moves the TCP non-linearly to the end position.

Example

SpotL p100, vmax, spot10, gun7, tool7;
SpotJ p100, vmax, spot10, gun7, tool7;

This is the only instruction needed to implement a complete welding operation.

The TCP for tool7 is moved on a linear or a non-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 closes in advance on it’s way to the position. The welding is started and supervised until finished and the gun is reopening.

Note. The program continues to execute after the weld has started and is not blocked until the next order that contains a robot motion. This may be inhibited by the switch \NoConc (see below).

Spotdata spot10 contains parameters to the welding equipment.

Gundata gun7 contains gun specific weld data.

Arguments


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.

Spot Data type: spotdata

Spot data that is associated with the weld process equipment.
InPos

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.

NoConc

The optional argument \NoConc prevents the program from continuing the execution of the following instruction until the actual weld is finished.

Retract

The optional argument \Retract will make the gun open to its large gap (retract) after the weld. If the argument is omitted the gun opens to its small gap (work).

Gun

Weld specific tool data for the gun in use.

Tool

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 of the tips when the gun is closed.

WObj

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.

Customizing

The SpotWare package gives the user plenty of scope for customising the SpotL/J instruction:

- by user defined data which affects the internal behaviour in SpotL/J. (See System Module SWUSRF, SWUSRC).
- by user defined routines which are called at predefined spots in the internal sequence. (See - System Module SWUSRF, SWUSRC).
- by changing the I/O configuration. (See - System Parameters SpotWare).

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

Internal sequence in a SpotL/J instruction:

- The gun starts to move towards the position.
- The gun pressure is set.
- The weld program number is set for the external weld timer.
- The preweld supervision is done.
- The gun starts to close at a defined time before the position (if argument \InPos is not used).
- The gun reaches the position.
- The start signal is sent to the weld timer.
- The weld counter is incremented.
- The program execution continues to the next movement instruction (unless argument \NoConc is used).
- When the ready signal from the timer is reached, the gun is opened.
- The postweld supervision is done.
- The robot moves to the next position.
Task 0

SpotL
- Call user routine `sw_close_time` and calculate the preclose time for the gun.
- Calculate the path to the programmed robot position.
- If motion is blocked wait for the motion to be released.
- The gun starts to move towards the position.
- The process start event is activated at motion start.
- The gun preclose event is activated at a defined time before the position. If the argument \InPos is used the event comes when the robot reaches the position.
- The robot reaches the position.
- The weld counter is incremented.
- Wait for process synchronisation, i.e. when the signal `inhib_move` is set to 0.

SpotL
- Call user routine `sw_close_time` and calculate the preclose time for the gun.
- Calculate the path to the programmed robot position.
- If motion is blocked wait for the motion to be released.
- The gun starts to move towards the position.
- The process start event is activated at motion start.
- The gun preclose event is activated at a defined time before the position. If the argument \InPos is used the event comes when the robot reaches the position.
- The robot reaches the position.
- The weld counter is incremented.
- Wait for process synchronisation, i.e. when the signal `inhib_move` is set to 0.

Task 1

main
- Initiate the process
- Wait

Trap
- Set the signals `proc_run` and `inhib_move`.
- Transfer the contents of the SpotL/J-parameters to the internal data.
- Call user routine `sw_set_pressure`.
- Set the weld program number if `sw_start_type` = 0.
- Wait for the gun preclose event.
- Call user routine `sw_preweld_sup`.
- Call user routine `sw_close_gun`.
- Wait for the robot to reach the position.
- The movement to the next position is blocked.
- Set the start signal to the weld timer.
- Reset the signal `inhib_move` (if not the argument \NoConc is used)
- Wait for the weld ready signal
- Reset the start signal.
- Call user routine `sw_open_gun`.
- Call user routine `sw_postweld_sup`.
- Reset the signals `proc_run` and `inhib_move` and release motion.

if not the \NoConc is used if the \NoConc is used
Figure 3 Spot weld sequence.

1. Fetch instruction and prepare motion
2. Block motion
3. Weld ready
4. Start weld
5. Set weld program no.
6. Gun open
7. Pressure OK
8. Set pressure
9. Gun closed
10. Close gun
11. Motion
12. Start weld
13. Weld ready
14. Block motion
15. Prepare motion
**Gun closure**

The gun closure is activated at a defined time before the weld position, irrespective of the actual speed. Closure shall be finished when the gun arrives at the end position. The close time is defined in the gun data. It relates to the time for closing the gun from the work stroke and is dependent on the chosen gun pressure (in spot data).

A maximum of four discrete gun pressures can be chosen in spot data. The gun pressure is set as soon as the motion towards the position starts. A check is made on whether or not the gun pressure has been attained, if so demanded by gun data.

**Welding**

The start signal is sent to the timer as soon as the robot has reached the end position. Before the start signal is sent, a number of supervisions must also be acknowledged. These are user defined routines.

The start signal is high during the entire welding period. It is reset either after weld ready or weld timeout.

When the option for program triggered timers is used the program number is set with the start signal and reset to zero after a successful weld or timeout.

**Gun opening**

The gun opens to a small or large stroke, after the welding has finished, depending on the parameter \( \text{Retract} \). The opening is supervised in such a way that a gun open signal is expected, if so demanded by gun data.

The gun is also opened after a weld error and other error situations.

**Motion**

To gain time the program pointer goes on to the next motion instruction while the welding is still performing. This makes it possible to perform the next motion immediately on a release order.

This means that instructions after \( \text{SpotL/J} \), other than motional, will be performed concurrently with the welding. If this is not the intention the parameter \( \text{NoConc} \) shall be used.

The end position is related to the used object coordinate system in \( WObj \).

**Customising**

There is a wide range of tools for customising the SpotWare package by offering RAPID user routines. These are straight forward routines hooked into the base software. The following functions can be programmed freely by the user:

Preclose time calculation - \( \text{sw\_close\_time} \) (defined in \( \text{SWUSRF} \))
Pressure setting - sw_set_pressure (defined in SWUSRC)

Preweld supervision - sw_preweld_sup (defined in SWUSRC)

Gun closure - sw_close-gun (defined in SWUSRC)

Gun opening - sw_open-gun (defined in SWUSRC)

Postweld supervision - sw_postweld_sup (defined in SWUSRC)

Error recovery - sw_error_recover (defined in SWUSRC)

Note that the default code gives the same behaviour as described in the SpotWare documentation.

---

**Program stop and restart**

*Program stop during the motion and restart*

The robot stops on the path. If the signal to close the gun has already been sent the gun reopens at the stop.

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

*Program stop during welding and restart*

The welding is finished. The validation of the weld is done after the stop and the gun opens. If \NoConc was not used, the program pointer will leave the current instruction and will point to the next instruction containing a robot motion.

---

**Instruction by instruction execution**

*Forwards*

The motion and the welding are done.

*Backwards*

The gun is set to work or retract stroke depending on \Retract. The motion is performed backwards.
Simulated welding

**Full simulation of a timer.**

Activated by setting `sw_sim_weld` TRUE. This will inhibit the start signal to the timer. The simulation time is defined in `sw_sim_time`.

**Simulation in the timer**

Activated by setting `sw_inhib_weld` TRUE. This will set the `current_enable` signal low at the next weld.

Error handling

**Events in an error situation**

When SpotL/J is stopped by a supervision, the following take place:

- The signal `process_error` is set.
- An error message is displayed.
- The error message is logged.

**Error situations**

The following error situations can occur:

- Instruction parameter error.
- Supervision error before welding.
- Supervision error for gun closure.
- Weld error.
- Supervision error for gun opening.
- Supervision error after welding and before the next motion.

All the supervision behaviour described above, except parameter error and weld error, can be modified by the user.

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

The behaviour below is programmed as default.
Instruction parameter supervision

The error occurs when SpotL/J is called with faulty parameters. The robot moves to the programmed position with a MoveL/J.

The parameter must be changed and the current instruction must be restarted from the beginning.

Supervision before welding

Supervisions in the user defined routine sw_preweld_sup are executing. (See - System Module SWUSRC).

Weld error

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

- The start signal is set low.
- The gun opens.
- If no automatic reweld the both signals weld_error and process_error sets high.
- The following manual choices are available:
  
  Automatic mode: Service / Reweld
  
  Manual mode: Service / Skip / Reweld (see the dialog box in Figure 4).

```
<table>
<thead>
<tr>
<th>Program Waiting for Data!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weld_ready timeout</td>
</tr>
<tr>
<td>Service        Skip       Reweld</td>
</tr>
</tbody>
</table>
```

Figure 4  Dialog box for weld error.
**Restart by choice Service**

This is not available in execution mode “stepwise forward”

- The signal `process_error` is reset.
- The user defined routine for weld fault `sw_service_wf` is executing, for instance moving to a home position.
- The robot moves back to the spot welding point after user confirmation.
- Back to the dialog as shown in Figure 4.

**Restart by choice Skip**

- The signal `process_error` is reset.
- The signal `reset_fault` is pulsed.
- The signal `weld_error` is reset.
- The program execution is resumed but omitting the faulty weld.

**Restart by choice Reweld**

- The signal `process_error` is reset.
- The signal `reset_fault` is pulsed.
- The program execution is set in a time delay of `sw_reset_time2`.
- The user routines `sw_set_pressure`, `sw_preweld_sup` and `sw_close_gun` will be called before the start signal is set.

**Restart by manual action SwRunProc**

- Same result as Reweld.

**Supervision after the weld**

Supervisions in the user defined routine `sw_postweld_sup` are executing. (See - System Module SWUSRC).

**Gun closure supervision**

Supervisions in the user defined routine `sw_close_gun` are executing. The error occurs if the chosen pressure is not reached after a certain time. The signal `p1_ok` is supervised
if \( \text{gun}_\text{pressure} = 1 \) i spotdata etc.

- The user routine \( \text{sw}\_\text{open}\_\text{gun} \) is called.
- The signal \( \text{process}\_\text{error} \) is set high.
- The following manual choices is available: \text{Service / Retry}.
- Choice \text{Service}: first the signal \( \text{process}\_\text{error} \) is reset and then the user defined routine \( \text{sw}\_\text{service}\_\text{cg} \) is executing.
- Back to the manual choice: \text{Service / Retry}.
- Choice \text{Retry}: the user routine \( \text{sw}\_\text{close}\_\text{gun} \) is called and the pressure is tested again.

\textbf{Gun opening supervision}

Supervisions in the user defined routine \( \text{sw}\_\text{open}\_\text{gun} \) are executing. The error occurs if the gun has not opened after a certain time. The signal \( \text{tip1}\_\text{open} \) or \( \text{tip1}\_\text{open} \) and \( \text{tip2}\_\text{open} \) if \( \text{nof}_\text{tips} = 2 \) in gundata is supervised.

- The signal \( \text{process}\_\text{error} \) is set high.
- The following manual choice is available: \text{Service / Retry}
- Choice \text{Service}: first the signal \( \text{process}\_\text{error} \) is reset and then the user defined routine \( \text{sw}\_\text{service}\_\text{cg} \) is executing.
- Back to the dialog shown in Figure 4.
- Choice \text{Retry}: the user routine \( \text{sw}\_\text{open}\_\text{gun} \) is called and the signals are tested again.

\textbf{User defined error recovery}

When user defined error recovery is switched on \( (\text{sw}\_\text{user}\_\text{recover} = \text{TRUE}) \), the routine \( \text{sw}\_\text{error}\_\text{recover} \) is called if any of the above error cases occur, except for parameter error. The input parameters carry information about the error case and the chosen error text.

This routine allows customising the error handling response, i.e. the teach pendant layout and how to resume. See Predefined Data and Programs - \text{System Module SWUSRC}.

Note: no warning will be stored in the robot error message log.

\textbf{Manual actions}

Manual actions allow the execution of spot weld functions without having to program a SpotL/J-instruction. It can be used as a tool to test user defined code before programming the line program.

The manual actions execute the same process code as when running SpotL/J, i.e. full
error recovery etc. is provided.

Manual actions execute in any system state except when a SpotL/J-instruction is in progress. Manual actions are activated by assigned virtual outputs.

See - Spot Welding.

Communication

SpotL/J communicates with its equipment using parallel signals.

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

Some weld timers with serial interface are supported. In those cases the SpotL/J still faces a parallel interface and the serial communication messages are mapped internally. See separate documentation.

Power failure handling

At system restart after power failure:

- All spotweld output signals are set to the old status, as well as gun close signals and weld start signals.

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.

Syntax

SpotL or SpotJ

[ 'ToPoint ':=’ ] < expression (IN) of robtarget > ‘,’
[ 'Speed ':=’ ] < expression (IN) of speeddata > ‘,’
[ 'Spot:':=’ ] < persistent (PERS) of spotdata >
[ '\' InPos ]
[ '\' NoConc ]
[ '\' Retract ] ‘,’
[ 'Gun' :=’ ] < persistent (PERS) of gundata > ‘,’
[ 'Tool ' :=’ ] < persistent (PERS) of tooldata >
[ '\' WObj :=’ ] < persistent (PERS) of wobjdata > ] ‘;’
## Related information

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other positioning instructions</td>
<td>RAPID Summary - <em>Motion</em></td>
</tr>
<tr>
<td>Definition of velocity</td>
<td>Data types - <em>speeddata</em></td>
</tr>
<tr>
<td>Definition of zone data</td>
<td>Data types - <em>zonedata</em></td>
</tr>
<tr>
<td>Definition of tool</td>
<td>Data types - <em>tooldata</em></td>
</tr>
<tr>
<td>Definition of work objects</td>
<td>Data types - <em>wobjdata</em></td>
</tr>
<tr>
<td>Definition of spot data</td>
<td>Data types - <em>spotdata</em></td>
</tr>
<tr>
<td>Definition of gun data</td>
<td>Data types - <em>gundata</em></td>
</tr>
<tr>
<td>Overview Spot welding</td>
<td>RAPID Summary - <em>Spot Welding</em></td>
</tr>
<tr>
<td>Customising tools</td>
<td>Predefined data and programs - <em>System Module SwUsrF, SwUsrC</em></td>
</tr>
<tr>
<td>I/O configuration</td>
<td>System parameters - <em>Spot Welding</em></td>
</tr>
<tr>
<td>Motion in general</td>
<td>Motion and I/O Principles</td>
</tr>
<tr>
<td>Coordinate systems</td>
<td>Motion and I/O Principles - <em>Coordinate systems</em></td>
</tr>
</tbody>
</table>
The system module SWUSRC contains data and routines aimed for customising the behaviour of the SpotWare application. They are used and executed commonly by all RAPID tasks of SpotWare.

NB. Any change of routines in SWUSRC requires the following three steps to affect the application (see also System Parameters Multitasking):

- save to HOME directory
- make a PSTART to affect the application

The names are predefined and used internally when a SpotL/J instruction is used. They must therefore not be changed.

Contents

Data

The following global data are predefined:

<table>
<thead>
<tr>
<th>Name</th>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sw_weld_counter</td>
<td>PERS num</td>
<td>Counter for welded spots. The counter is automatically incremented in the SpotL/J instruction.</td>
</tr>
<tr>
<td>sw_weld_counter := 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sw_gp_timeout</td>
<td>PERS num</td>
<td>Gun pressure timeout [s]</td>
</tr>
<tr>
<td>sw_gp_timeout := 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sw_wr_timeout</td>
<td>PERS num</td>
<td>Weld ready timeout [s]</td>
</tr>
<tr>
<td>sw_wr_timeout := 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sw_go_timeout</td>
<td>PERS num</td>
<td>Gun open timeout [s]</td>
</tr>
<tr>
<td>sw_go_timeout := 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sw_prog_max</td>
<td>PERS num</td>
<td>Max. value for prog_no in spotdata.</td>
</tr>
<tr>
<td>sw_prog_max := 63</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sw_inhib_weld</td>
<td>PERS bool</td>
<td>Flag to set the weld timer in simulation mode. If TRUE: The signal current_enable is cleared when the next SpotL/J instruction is executed.</td>
</tr>
<tr>
<td>sw_inhib_weld := FALSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
System Module SWUSRC

<table>
<thead>
<tr>
<th>Name</th>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sw_sim_weld</td>
<td>PERS bool</td>
<td>Flag to internally simulate the weld timer. If TRUE: Weld start signal is not activated.</td>
</tr>
<tr>
<td></td>
<td>sw_sim_weld := FALSE</td>
<td></td>
</tr>
<tr>
<td>sw_spot_id_req</td>
<td>PERS bool</td>
<td>Flag to request the spot identity of the programmed spots. If TRUE: It’s possible to fetch the current spot identity (the spot parameter name of the current spotdata) using the function SwGetCurrSpotID. If the serial link for Bosch PSS 5000 is in use, the weld programs will be requested by the spotdata name instead of using the program number.</td>
</tr>
<tr>
<td></td>
<td>sw_spot_id_req := FALSE</td>
<td></td>
</tr>
<tr>
<td>sw_sim_time</td>
<td>PERS num</td>
<td>Simulated weld time if sw_sim_weld = TRUE.</td>
</tr>
<tr>
<td></td>
<td>sw_sim_time := 0.5</td>
<td></td>
</tr>
<tr>
<td>sw_aut_reweld</td>
<td>PERS num</td>
<td>Number of automatic tries to reweld after weld ready timeout.</td>
</tr>
<tr>
<td></td>
<td>sw_aut_reweld := 0</td>
<td></td>
</tr>
<tr>
<td>sw_parity</td>
<td>PERS num</td>
<td>0 = none, 1 = odd, 2 = even</td>
</tr>
<tr>
<td></td>
<td>sw_parity := 0</td>
<td></td>
</tr>
<tr>
<td>sw_reset_time1</td>
<td>PERS num</td>
<td>Reset pulse [s]</td>
</tr>
<tr>
<td></td>
<td>sw_reset_time1 := 0.5</td>
<td></td>
</tr>
<tr>
<td>sw_reset_time2</td>
<td>PERS num</td>
<td>Wait time after the reset pulse and after each time current_enable is changed [s].</td>
</tr>
<tr>
<td></td>
<td>sw_reset_time2 := 0.5</td>
<td></td>
</tr>
<tr>
<td>sw_async_weld</td>
<td>PERS num</td>
<td>TRUE means the start of the weld is set independently of inpos, i.e. immediately after closing the gun.</td>
</tr>
<tr>
<td></td>
<td>sw_async_weld := FALSE</td>
<td></td>
</tr>
<tr>
<td>sw_servo_corr</td>
<td>PERS num</td>
<td>Correction time for internal delay. This value affects the preclosing time. The correct time which is dependent on motion configuration is automatically set at start up.</td>
</tr>
<tr>
<td></td>
<td>sw_servo_corr := 0.061</td>
<td></td>
</tr>
</tbody>
</table>
The following predefined data is used by the manual actions to operate on:

<table>
<thead>
<tr>
<th>Name</th>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sw_start_type</td>
<td>PERS num sw_start_type:=0</td>
<td>Tells the system how to start the weld process in the timer:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: The weld process start is triggered by a specific start signal (e.g. start1). The program number is set in advance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: The weld process is triggered by the program number outputs. No presetting of the program number</td>
</tr>
<tr>
<td>sw_user_recover</td>
<td>PERS num sw_user_recover := FALSE</td>
<td>TRUE means the user function sw_error_recover is called in case of a process error. No warning will be stored in the robot error message log. FALSE means standard error recovery is used.</td>
</tr>
</tbody>
</table>

The following predefined data is used by the manual actions to operate on:

<table>
<thead>
<tr>
<th>Name</th>
<th>Declaration</th>
</tr>
</thead>
<tbody>
<tr>
<td>man_gun</td>
<td>PERS gundata man_gun := [1, 4, TRUE, TRUE, 0, 0, 20000, 20000, 0.050, 0.050, 0.050, 0.050, 0, 0, 0, 0, 0]</td>
</tr>
<tr>
<td>man_spot</td>
<td>PERS spotdata man_spot := [1, 1, 1, 1];</td>
</tr>
<tr>
<td>man_retr</td>
<td>PERS num man_retr := 0;</td>
</tr>
</tbody>
</table>

**Routines**

The following predefined routines are installed with the application. They are all used by the SpotLJ instruction.

These routines have a default functionality but can easily be changed.
User defined supervision routines for SpotL/J called from task SW_PROC:

The following routines are called by the SpotL/J sequence.

PROC sw_preweld_sup ()

spotweld preweld supervision

The routine is executed before each start of the weld process.

If an error text is assigned to the text string errtext, then the system is automatically stopped with the error text on the display and with a possibility to restart the supervision routine and continue the program execution.

Default functionality:
If the simulate weld or inhibit weld functions are not activated, then the following digital input signals are tested:

timer_ready
flow_ok
temp_ok
current_ok

In the event of an error, a suitable error text is assigned to the text string errtext.

If the signal timer_ready = 0, then one attempt to reset the timer is made and a new check of the input is made before an error is indicated.

PROC sw_postweld_sup ()

spotweld postweld supervision

The routine is executed after the weld process before movement to the next position.

If an error text is assigned to the text string errtext, then the system is automatically stopped with the error text on the display and with a possibility to restart the supervision routine and continue the program execution.

Default functionality:
None

PROC sw_close_gun ()

spotweld close gun

The routine is executed each time a closing of the gun is ordered. N.B. also the manual action.

If an error text is assigned to the text string errtext_close, then the system is automatically stopped with the error text on the display and with a possibility to restart the supervision routine and continue the program execution.

Default functionality:
See - SpotL/J.
**PROC sw_open_gun (num context)**

spotweld open gun

The routine is executed each time an opening of the gun is ordered. N.B. also the manual action.

Parameters:

- **context**: reason for opening. A negative value (-1) is used to leave out the gun open tests and to ignore the `errtext_open` when a quick open is needed in an error situation. The situations are:
  - Supervision error before welding.
  - Supervision error for gun closure.
  - Weld error.
  - Supervision error after welding.
  - When the signal `SwSkipProc` is set or if a `Skip` has been chosen.

If an error text is assigned to the text string `errtext_open`, then the system is automatically stopped with the error text on the display and with a possibility to restart the supervision routine and continue the program execution.

Default functionality:
See - *SpotL/J*.

**PROC sw_set_pressure ()**

spotweld set pressure

The routine is executed in the preparation phase of the spot weld process.

Default functionality:
Sets the pressure output group according to `gun_pressure` in spotdata. See - *SpotL/J*.

**FUNC num sw_error_recover (num error_type, string err_text)**

spotweld error recover

The function is called in an error situation if the variable `sw_user_recover` is set TRUE.

Parameters:

- **error_type**: type of error that occurred. Possible cases are:
  - SW_WELD_ERR: weld error timeout
  - SW_OG_ERR: open gun error. Error reported by `sw_open_gun`
  - SW.CG_ERR: close gun error. Error reported by `sw_close_gun`
  - SW_PRESUP_ERR: preweld supervision error. Error reported by `sw_preweld_sup`
  - SW_POSUP_ERR: postweld supervision error. Error reported by `sw_postweld_sup`
• **err_text**: text string that was returned by the function that reported the error. In case of a weld error it is the standard “weld ready timeout” message.

The return value of this function defines how the SpotWare shall resume after this error. There are four possible return values:

- **SW_RETRY**: the action that produced the error is re-executed.
- **SW_CANCEL**: the current spot weld process is abandoned and cleaned up.
- **SW_SERVICE**: the service routine corresponding to the error produced is executing. Note that no service routines exist for preweld and postweld supervision.
- **SW_REWELD**: the weld process is started from the beginning after weld error and after errors reported by:
  - `sw_open_gun`,
  - `sw_close_gun`,
  - `sw_preweld_sup`,
  - `sw_postweld_sup`.

**User defined independent supervision routines called from task SW_SUP:**

The following routines are called independently of the SpotL/J-sequence.

**PROC sw_sup_init ()**

spotweld supervision init

The routine is executed at warm start. Here the user interrupt numbers used by `sw_sup_trap` should be initialised.

Default functionality:
Connection and activation of signals and interrupts used by `sw_sup_trap`.

**PROC sw_motor_on ()**

spotweld motor on

Routine called by the trap `sw_sup_trap`.

Default functionality:
Connected to interrupt number `imotor_on`. Set the signal `weld_power` and pulse the signal `water_start`.

**PROC sw_motor_off ()**

spotweld motor off

Routine called by the trap `sw_sup_trap`.

Default functionality:
Connected to interrupt number `imotor_off`. Reset the signal `weld_power`. 
PROC sw_proc_ok ()

spotweld process ok

Routine called by the trap sw_sup_trap.

Default functionality:
Connected to interrupt number iproc_ok. Set the signal weld_power and pulse the signal water_start.

PROC sw_proc_error ()

spotweld process error

Routine called by the trap sw_sup_trap.

Default functionality:
Connected to interrupt number iproc_error. Reset the signal weld_power.

PROC sw_sup_curren ()

spotweld supervise current enable

Routine called by the trap sw_sup_trap.

Default functionality:
Connected to interrupt number icurr_enable. Set the signal weld_power and pulse the signal water_start.

PROC sw_sup_curdis ()

spotweld supervise current disable

Routine called by the trap sw_sup_trap.

Default functionality:
Connected to interrupt number icurr_disable. Reset the signal weld_power.

User defined trap routines:

The interrupts in these trap routine will always be alert to execute independently of the actual system state.

TRAP sw_sup_trap ()

spotweld supervision trap

Trap routine connected to the supervision.

Default functionality:
Control of the output signals water_start and weld_power depending on the system state. See - System Parameters SpotWare.
System Module  SWUSRF

The system module SWUSRF contains data and routines aimed for customising the behaviour of the SpotWare application. They are used and executed by the RAPID foreground task of SpotWare.

The routine names are predefined and used internally when a SpotL/J instruction is used. They must therefore not be changed.

Contents

Data

The following global data are predefined:

<table>
<thead>
<tr>
<th>Name</th>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sw_inpos</td>
<td>PERS bool sw_inpos := FALSE</td>
<td>If TRUE: Gun preclosing is deactivated.</td>
</tr>
<tr>
<td>sw_close_corr</td>
<td>VAR num sw_close_corr := 100</td>
<td>Correction factor used when the next gun preclose time is calculated [%]. Is automatically reset to 100 after each preclose calculation. Preferably this data can be changed in a separate assignment instruction just before the influenced SpotL/J instruction.</td>
</tr>
</tbody>
</table>

The following variables are predefined but their names and values can be changed freely.

<table>
<thead>
<tr>
<th>Name</th>
<th>Declaration</th>
</tr>
</thead>
<tbody>
<tr>
<td>gun1</td>
<td>PERS gundata gun1 := [1, 4, TRUE, TRUE, 0, 0, 20000, 20000, 0.050, 0.050, 0.050, 0, 0, 0, 0]</td>
</tr>
<tr>
<td>toolg1</td>
<td>PERS tooldata toolg1 := [TRUE, [[0, 0, 0], [1, 0, 0, 0]], [-1, [0, 0, 0],[1, 0, 0, 0], [0, 0, 0]]</td>
</tr>
<tr>
<td>spot1</td>
<td>PERS spotdata spot1 := [1, 1, 1, 1]</td>
</tr>
</tbody>
</table>
Routines

The following predefined routines are installed with the application. They are all used by the *SpotL/J* instruction.

These routines have a default functionality but can easily be changed.

**User defined service routines:**

After execution of a service routine, the robot moves to the interrupted weld position with reduced speed. The text “Service routine ready” and the old function keys are shown.

**PROC sw_service_cg ();**

```c
spotweld service close gun
```

The routine is executed when the gun error “Gun pressure not reached” occurs and the function key “Service” is pressed.

By pressing Retry back from the service routine, a new attempt to close the gun is made and then program execution continues.

Default functionality:
No functions, only an information text and a “Return” key are shown. The service routine is ended when “Return” is pressed.

**PROC sw_service_og ();**

```c
spotweld service open gun
```

The routine is executed when the gun error “Gun open timeout” occurs and the function key “Service” is pressed.

By pressing Retry back from the service, a new test will be made if the gun is open, and then program execution continues.

Default functionality:
No functions, only an information text and a “Return” key are shown. The service routine is ended when “Return” is pressed.

**PROC sw_service_wf ();**

```c
spotweld service weld fault
```

The routine is executed when the error “Weld ready timeout” occurs and the function key “Service” is pressed.

By pressing Reweld back from the service, a new attempt to weld the spot is made and then program execution is continued.
Default functionality:
No functions, only an information text and a “Return” key are shown. The service routine is ended when “Return” is pressed.

General user routines and functions:
The following routines are connected to different entries in the SpotWare RAPID base software.

FUNC num sw_close_time (spotdata spot, gundata gun);
   spotweld close time calculation

The function is executed during motion preparation. It returns the preclosing time for the gun.

Input parameters:
spot of type spotdata: actual spot given in the SpotL-instruction in preparation
gun of type gundata: actual gun given in the SpotL-instruction in preparation

Default functionality:
The resulting time is dependent on the four pressure levels in gundata.
System Module  

SWTOOL

The system module SWTOOL contains data, functions and routines. The module is declared NOVIEW and contains utilities to be used as a toolbox when customising SpotWare. This module is accessible commonly by all RAPID tasks of SpotWare.

Contents

Function return values

- SW_OK
- SW_RETRY
- SW_SERVICE
- SW_ERROR
- SW_TIMEOUT
- SW_CANCEL
- SW_REWELD

User error recovery types (see function sw_user_recover in SWUSRC)

- SW_WELD_ERR
- SW(CG)_ERR
- SW_OG_ERR
- SW_PRESUP_ERR
- SW_POSUP_ERR
Routines

The following predefined routines are installed with the application. They are used to influence SpotL/J instruction.

**PROC SwSetCurrSpot (spotdata spot)**

SpotWeldSetCurrentSpot.

The function changes the spotdata parameter of the running spotware process.

**PROC SwSetCurrGun (gundata gun)**

SpotWeldSetCurrentGun

The function changes the gundata parameter of the running spotware process.

**PROC SwSetCurrRetr (num retr)**

SpotWeldSetCurrentRetract

The function changes the Retract switch parameter of the running spotware process.

- retr 0: work stroke
- retr 1: retract stroke

**PROC SwGetCurrNoConc (bool noconc)**

SpotWeldSetCurrentNoConcurrency

The function changes the NoConc switch parameter of the running spotware process.

- noconc TRUE: NoConc is present
- noconc FALSE: NoConc is not present

**PROC SwSetCurrSpotID (string spot_id)**

SpotWeldSetCurrentSpotIdentity

The function changes the spotdata parameter name of the running spotware process.
Functions

The following predefined functions are installed with the application.

**FUNC spotdata SwGetCurrSpot**

SpotWeldGetCurrentSpot

The function returns the content of the spotdata parameter of the running spotware process.

**FUNC gundata SwGetCurrGun**

SpotWeldGetCurrentGun

The function returns the content of the gundata parameter of the running spotware process.

**FUNC num SwGetCurrRetr**

SpotWeldGetCurrentRetract

The function returns the content of the Retract switch parameter of the running spotware process.

Return value 0: work stroke

Return value 1: retract stroke

**FUNC bool SwGetCurrNoConc**

SpotWeldGetCurrentNoConcurrency

The function returns the content of the NoConc switch parameter of the running spotware process.

Return value TRUE: NoConc is present

Return value FALSE: NoConc is not present

**FUNC string SwGetCurrSpotID**

SpotWeldGetCurrentSpotIdentity

The function returns the spotdata parameter name of the running spotware process.
**System Module SWTOOL**

**SpotWare**

**FUNC num SwWaitInput (VAR signaldi input, num value \ num MaxWait)**

SpotWeldWaitInput

Waits until the signal *input* has reached *value*. A max. time *MaxWait* can be added as an optional parameter. When running a user routine called by the spot weld process, in which the process is supposed to wait for an input signal, this function is used to let the system abandon the current process.

Return value SW_OK: Signal was set to output *value*.

Return value SW_TIMEOUT: The time specified in *MaxWait* has been exceeded.

Return value SW_CANCEL: The SpotWare process has received an abort and wants to cancel the current process. It shall cause a return from the current user routine.

**FUNC num SwWaitOutput (VAR signaldo output, num value \ num MaxWait)**

SpotWeldWaitOutput

Waits until the signal *output* has reached *value*. A maximum time *MaxWait* can be added as an optional parameter. When running a user routine called by the spotweld process, in which the process is supposed to wait for an output signal, this function is used to let the system abandon the current process.

Return value SW_OK: Signal was set to *value*.

Return value SW_TIMEOUT: The time specified in *MaxWait* has been exceeded.

Return value SW_CANCEL: The SpotWare process has received an abort and wants to cancel the current process. It shall cause a return from the current user routine.

**FUNC num SwErrorAck (VAR num input_key, string alert, string ktxt1, string ktxt2, string ktxt3, string ktxt4, string ktxt5)**

SpotWeldErrorAcknowledge

This function is a modified TPReadFK which can be aborted by the SpotWare kernel. The parameters are consistent with the standard TPReadFK. When this routine is not used for teach pendant dialogs, the SpotWare is hung up until released from the TP. The function is used by sw_error_recover in the module SWUSRC.

Return value SW_OK: normal execution, the result of the pushing a function key resides in *input_key*.

Return value SW_CANCEL: the spot weld process was aborted. Return from the calling user routine.
G
gundata 3, 15

M
manual gun control 4

P
programming
   spot welding 3

S
simulated spot welding 28
spot weld data 3
spot weld gun data 15
spot weld instructions 3
spot welding 21
   system parameters 9
spotdata 3, 19
SpotL 3, 21