HELLER

Operator Information

Integrated Process Monitoring IPM, SINUMERIK 840D sl

BI.000225-EN-03

Contract data	
Designation	Machining centre
Machine type	Integrated Process Monitoring IPM,
Control	SINUMERIK 840D sl

Gebr. HELLER Maschinenfabrik GmbH

Postfach 14 28
D-72604 Nürtingen

Neuffener Straße 54
D-72622 Nürtingen

Telefon: +49-(0)70 22-77-0
Telefax: +49-(0)70 22-77-610
E-Mail: sales@hnt.heller-machines.com
Internet: http://www.heller-machinetools.com

HELLER

Contents

CHAPTER 1

Integrated Process Monitoring IPM (option)

1 Integrated Process Monitoring IPM (option)

1.1 Scope of the Operator Information

Operator Information

This Operator Information supplements the Operator Manual and the Programming Instructions. It describes how to operate the "Integrated Process Monitoring" option.

Intended readership

The Operator Information is intended for the user (owner), the programmer and for the machine operator. It must be made available to this group of persons.

1.2 Integrated Process Monitoring IPM (option)

The Integrated Process Monitoring (IPM) is based in part on the work of the Qwt project.

1.2.1 Introduction

Standard procedure

The existing monitoring procedures disable a fitted tool if the actual monitored values of a cutting tip exceed their limit values. Possible limit values include maximum tool life, number of items or a maximum value for entering the tool length wear. The real tool condition cannot however be detected.

Integrated Process Monitoring (IPM)

Integrated Process Monitoring (IPM) is a dynamic addition to the existing standard procedure in connection with tool management.

Monitoring methods

IPM allows you to detect tool overloads early, thereby avoiding tool fracture and all the consequential damage to machine and tool. To achieve this aim, the IPM area, with which freely parameterisable monitoring methods can be generated, has been developed.

Sensors

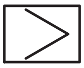
The signal sources - also known as sensors - are the digital drive technology variables: setpoint torque value, actual current value, effective power, feed force, axial and radial force. Drive signals are subject to a signal adjustment, the acceleration, friction and stoppage forces being eliminated so that the actual processing variable is available.

Tool-specific monitoring

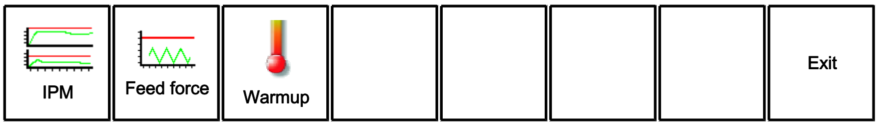
In a subsequent step, the specified monitoring methods must be assigned to the real tools (cutting tips) of the machine. This assignment takes place within the parameter / tool wear area.

1.2.2 Calling up Integrated Process Monitoring (IPM)

☒ Call up the machine's basic menu using the *Data menu key*.



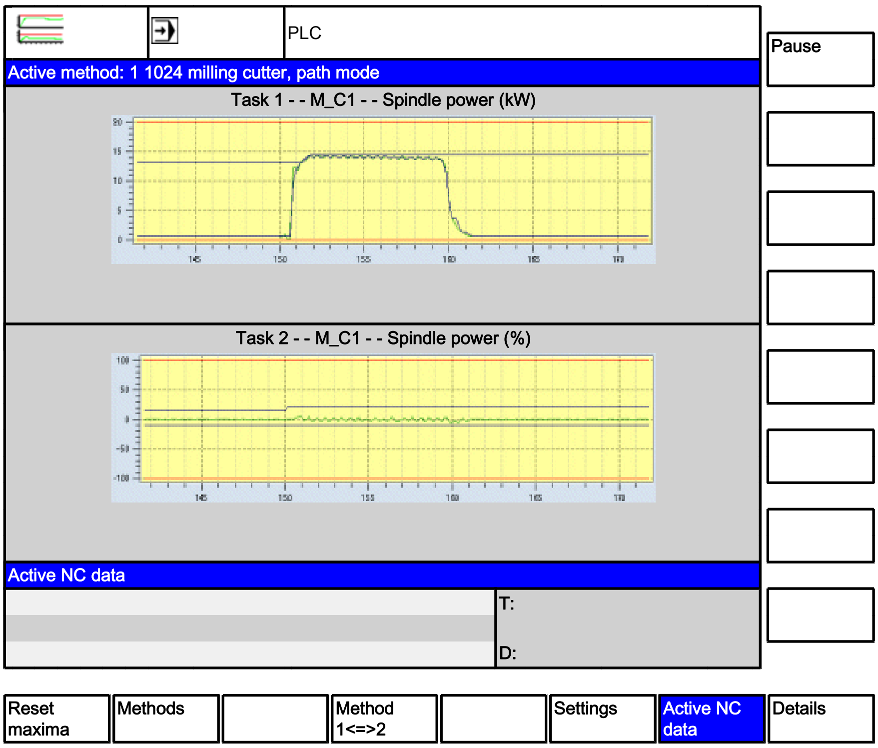
☒ Press the *etc.* key.
↳ The "IPM" softkey is displayed.



IPM

☒ Press the "IPM" softkey.

↳ The "IPM" basic screen is opened.
↳ The "Active NC data" are displayed.



Details

☒ To show a numerical display for a graphic illustration of the sensor values, press the "Details" softkey.

↳ The following values are shown in the numerical display:
- Upper limit
- Maximum value
- Minimum value
- Sensor value
- Lower limit
- Block average

ℹ You can control the display of the individual values in the curve by selecting and deselecting the corresponding checkboxes.

Selecting monitoring events

Settings

The events to be monitored can be activated in the "Settings" window area of the basic display:

- Tool break (with cutting and missing monitoring)
- Tool overload (with blunt monitoring)

For the monitoring operation, at least one event must be selected via the check field (tick).

☑ The machining program stops.

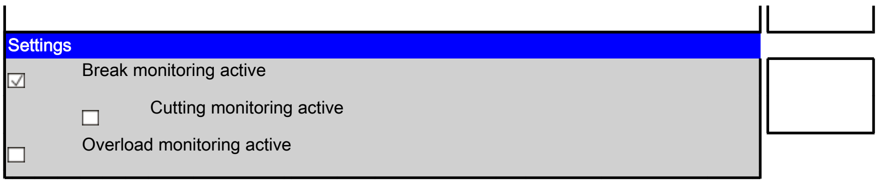
☑ The "IPM" basic display is opened.

Settings

☒ Press the "Settings" softkey.

↳ The "Settings" window area in the basic display opens.
The check fields show the current monitoring mode, e.g.:

- Break monitoring activated.
- Cutting monitoring deactivated.
- Overload monitoring deactivated.



Activating further monitoring events

☑ The "fracture monitoring active" line is highlighted.

☒ Move the highlight to the "overload monitoring active" using the *Cursor keys*.

☒ Press the *Highlight* key.

↳ A tick will appear in the check field to confirm.

↳ During subsequent machining operations, the tool is also monitored for overload situations.

Active cutting monitoring:

Cutting monitoring involves the lower limit of a break method being recorded. This ensures that a missing or broken tool is detected. Since cutting monitoring is a sub-function of break monitoring, it remains ineffective even when tool monitoring is switched off.

1.2.2.1 Displaying monitoring data

1 spindle - 2 monitoring methods

A maximum of two monitoring methods per tool cutting tip can be recorded and displayed:

- Break method and
- overload method.

The monitoring data for the two methods can be displayed alternately on the screen.

Depiction

The monitoring data for the current tool are displayed in graphical form in the window area.

The displayed method can contain a maximum of two tasks. In this case, two curves are displayed. The colour of the display alternates between yellow (display only) and green (alarm reaction active).

Pressing the "Details" softkey shows a numerical display for the graphic display.

Display monitoring data for method 1 (break method)

Preconditions

☑ A break method in the tool management system has been assigned to the current tool.

☑ The machine is in Automatic Mode. The current NC block runs with the tool to be monitored.

☑ The "IPM" basic display is opened.

☒ Activate break monitoring in the "Settings" window are of the basic screen.

↳ A maximum of two monitoring curves are displayed.
The curves contain the following data:

- Header: Monitoring method code with methods identification
- Red line: top or bottom monitoring limit
- Blue line: Minimum and/or maximum value
- Green curve: Current sensor value

i The maximum and the minimum of the current monitoring signal is denoted by the blue line. The minimum has a special function for the "fixed limit" process:
The value follows the current value and is "frozen" for recording the monitoring time of the cutting monitoring operating.

If the minimum value (blue) overruns or underruns the monitoring limit (red) during the recording period and/or at the recording time, an appropriate alarm response is triggered, if activated.

i If the function is switched off, the displayed curve disappears from the time signal diagram.

Switching over to Method 2 (overload method)

Precondition

☑ An overload method in the tool management system has been assigned to the current tool.

Method 1<=>2

☒ Press the "Method 1<=>2" softkey.

↳ Data from Monitoring Method 2 are superimposed.
The diagram for the overload method is shown instead of Method 1.

i If the "1<=>2 Method" softkey is pressed alternately, the monitoring results of both methods are displayed in rotation (toggle function). The detailed display is also switched.

1.2.2.2 Reset maximum values

It is often necessary to reset the maximum monitoring value, e.g.
- whilst an unknown tool is being recorded for the first time
- during severe interference

Reset maxima

☒ Press the "Reset maxima" softkey.
↳ The indicator of the displayed monitoring method skips to the current monitoring value.

1.2.3 Monitoring methods

General

Monitoring method

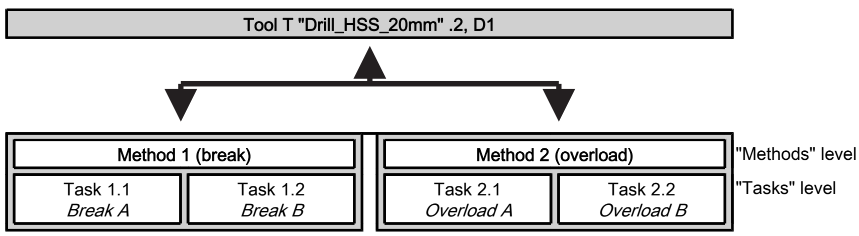
A monitoring method forms the basic unit for executing the tool monitoring system. The methods editor can be found in the integrated process monitoring IPM main menu
This editor contains all the information required for a monitoring event. The required signals, processes, tasks and operations must be specified.

Methods Editor

In the methods editor, the events to be monitored must be defined, whereby these can be recorded individually or in parallel.

Fracture and overload methods

Two monitoring methods can be assigned to each cutting edge. This enables a fracture event and an overload event to be recorded simultaneously. Two monitoring tasks per method can be defined, which are distinguished in terms of sensor, process, and evaluation algorithm etc.



Method templates

Default methods that are already completely parameterised can be requested for standard tool types. If you want to monitor similar tools, it is advisable to use the copy of a default method, which can be adapted to suit current conditions without the need for extensive editing.

1.2.3.1 Calling up the "Monitoring methods" window

Methods selection

The methods previously created are displayed in "Monitoring methods" on the left side of the window. Methods are clearly identified by their name and ID number.

The method ID number is of prime importance for internal control purposes.

Each of these is tagged with a coloured symbol to enable a quick distinction between break and overload method:
- Break method: Red setsquare.
- Overload method: Yellow setsquare.

To display further information and the tasks created for a monitoring method, you must select a monitoring method from the table using the cursor keys. The information will then be displayed on the right side of the window.

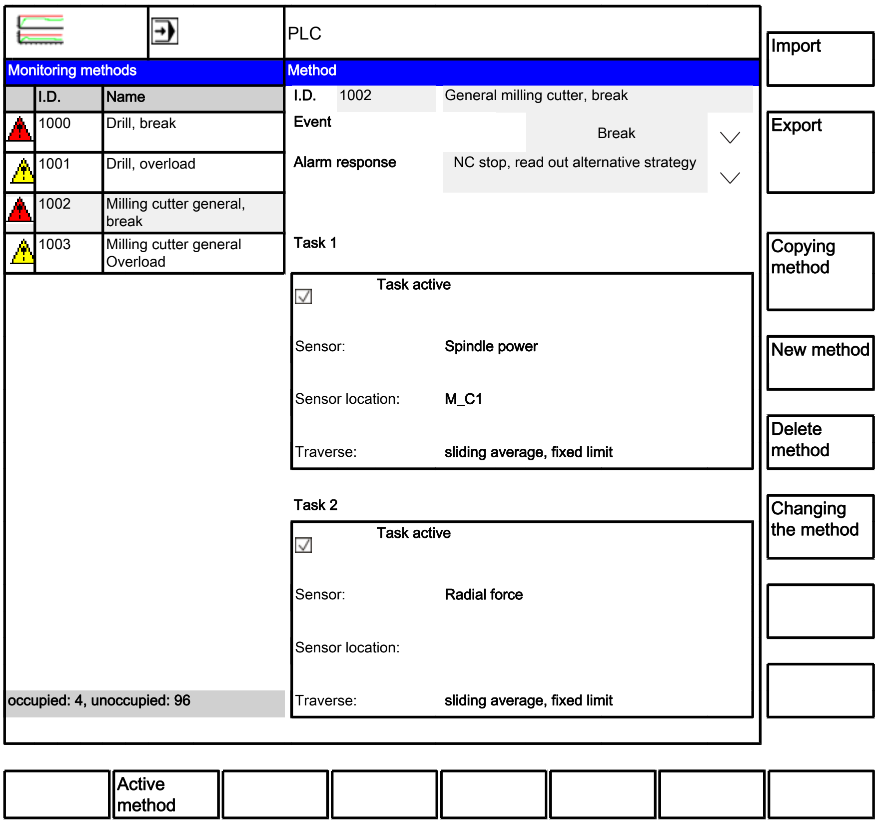
Preconditions

☑ The basic screen "Integrated Process Monitoring (IPM)" is open.

Methods

☒ Press the "Methods" softkey.

↳ The "Monitoring methods" window opens.



1.2.3.2 Description of monitoring method elements

Methods

The following input fields are available for defining the methods:
- ID and name
- Event
- Alarm response

ID and name

The system automatically suggests the next free ID.
You can define any method name. This is used to identify the method.

Event, alarm response

The event determines the monitoring task. An alarm response is permanently assigned, depending on the event you have selected.

Event	Alarm response
deactivated The method is not active	-

HELLER

Break The monitoring task is the break method.	**Enable NC stop and/or alternative strategy** When a break alarm occurs, all axes are stopped. The "alternative strategy" option is selected if present.
Overload The monitoring task is the overload method.	**Disable tool** When an overload alarm occurs, the affected tool is set to "disabled" at the next tool change.

Tasks

A maximum of two tasks can be defined for each method. The following input fields are available:

Task active

For check field is used for switching the current monitoring task off and on.

Sensor

All sensors available for parameterising the task are offered in a selection list:

- Torque (Nm)
- Current (amps)
- Spindle power (KW)
- Feed force (kN)
- Radial force (kN)
  - On G17: X/Y force component
  - On G18: Z/X force component
  - On G19: Y/Z force component
- Axial force (kN)
  - On G17: Z force component
  - On G18: Y force component
  - On G19: X force component

Sensor location

The axes feasible in respect of the set sensor are displayed here.

Process

The following signal processes are available for parametising the tasks:

Current value, fixed limit

The signal is directly compared to a fixed upper and lower limit, without further filtering.

Parameters:
- Upper limit
- Lower limit

Sliding average, fixed limit

The signal is averaged by the sliding method in which the old average and the current measured value are linked to form a new average. The mean is compared to a fixed upper and lower limit. This method has a smoothing effect, if the sensor mean factor is greater than 1.

Parameters:
- Upper limit
- Lower limit
- Average factor for sensor

Current value, live threshold

The signal is averaged by the sliding method in which the old average and the current sensor value are linked to form a new average.

This averaged value is used to derive the upper and lower limits values (thresholds), with which the unsmoothed signal is compared. The method is suitable only for detecting fractures.

Parameters:
- Upper limit
- Lower limit
- Average factor for sensor

Sliding average, fixed limit + missing/blunt

The signal is averaged by the sliding method in which the old average and the current sensor value are linked to form a new average. The average thus determined is then compared to a fixed upper and lower limit. This method has a smoothing effect if the "Average factor sensor" parameter is greater than 1.

"Blunt" and "missing" statuses can also be monitored.

"Blunt" monitoring:
Blunt (worn) tools can easily be detected in combination with the overload method. This is done by comparing the block average of the sensor signal to a top blunt threshold (parameter: missing or blunt limit). The limit must be determined empirically. This is done by comparing the block average of the sensor signal to a top blunt threshold (parameter: missing or blunt limit). The limit must be determined empirically.

"Missing" monitoring:
Missing tools can be effectively detected (similar to cutting monitoring) if the process is combined with the break method. This is done by comparing the block average of the sensor signal to a bottom missing threshold (parameter: missing or blunt limit). The limit must be determined empirically.

Parameters:
- Upper limit
- Lower limit
- Average factor for sensor
- Missing and/or blunt limit

Unless CS_IPMCO is called up, the "Incorrect or impassive limit" parameter has no effect.

Sliding average, live threshold

This process is similar to the "Current value, live threshold" process although in this case, the monitoring signal is pre-filtered. Both the parameters for the live threshold and also an "Average factor for sensor" parameter must be defined for noise suppression.

Parameters:
- Tolerance
- Switching point
- Average factor for threshold
- Average factor for sensor

Control override without Alarm

This method controls the path override within the permitted limits until the sensor signal reaches its setpoint value.

Parameters:
- Upper override limit
- Lower override limit
- Setpoint value

Control override with Alarm

This method controls the path override within the permitted limits until the sensor signal reaches its setpoint value. If during the control operation, the lower override limit is reached, an alarm is generated.

Parameters:
- Upper override limit
- Lower override limit
- Setpoint value

Input/activation

The following parameters can be selected for defining and scaling the monitoring diagrams: Depending on the type of process, only the sub-quantity of the parameter is available.

Upper limit

For the "actual value, fixed limit" and "sliding average, fixed limit" process (+ missing/blunt)" If the current value exceeds the upper limit, an alarm reaction is produced.

The upper limit must firstly be determined empirically through a test cut.
- Unit dependent on current sensor
- Floating point number

Lower limit

For the "current value, fixed limit" and "sliding average, fixed limit (+missing/blunt)" processes in combination with the break method. If at the time of cutting monitoring (programming command CS_IPMCO), the current value falls below the lower limit, an alarm response is triggered.

The lower limit must firstly be determined empirically through a test cut.
- Unit dependent on current sensor
- Floating point number

Average factor for sensor, average factor for threshold

Average factors must be defined for processes based on "Sliding average" and/or "Live threshold":
- Average factor sensor: For sliding average
- Average factor threshold: for live threshold

The signal or the thresholds can be steadied (smoothed) using the average factor. In principle: The higher the value, the greater the filter effect.
- Typical value: 5
- Integer: (1 .... 2^{31})

The display is steadied on the following settings:
- If processes based on "Fixed limit" involve the "Average factor for sensor" being increased.
- If processes based on "Live threshold" involve
  - the "Average factor for sensor" being increased
  - and/or the "Average factor for threshold" being reduced.

Tolerance [%]

Provided for the process
- "current value, live threshold" and
- "sliding average, live threshold".

An alarm reaction is generated.

If the current value and sliding average for the sensor signal overruns or underruns the thresholds derived from the tolerance value respectively, an alarm response is triggered. (Determination: See notes on "upper, lower threshold".)

The value for the tolerance input depends particularly on the machining process (rough or fine machining). If working with a lower average factor, a narrower tolerance range can also be selected. This enables the derived thresholds (upper and lower limit) to track the current value more quickly.

For unknown sensor behaviour, it is always advisable to make a test cut.
- Typical value: 25
- Unit %
- Integer

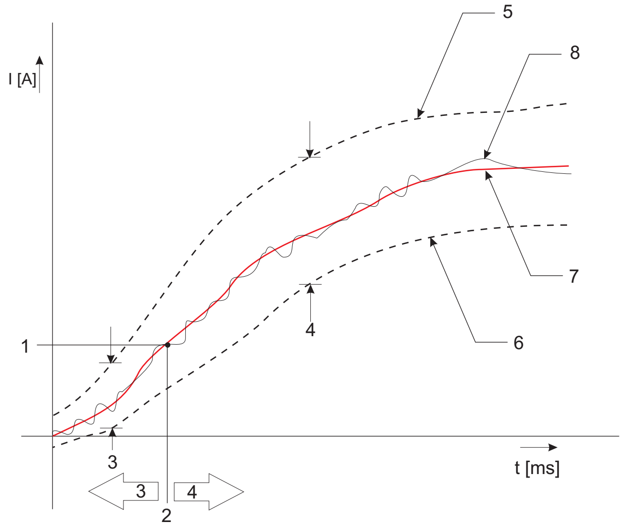
Switching point abs/% tol.

For the process based on "Live threshold".

The value assigned to the switching point is the sensor value determined from a test cut.

Depending on the level of the current sensor value, the upper and lower threshold values are determined in various ways.
- If the current sliding average is smaller than the switching point, a fixed tolerance hose is used. The upper and lower thresholds are calculated as percentages relative to the switching point.
- If the current sliding average is greater than the switching point, a variable tolerance hose is used. The upper and lower thresholds are calculated as percentages relative to the sliding average.

The graph below shows the detailed correlations.



1 Switching point
2 Tolerance loop
3 fixed
4 variable
5 upper threshold
6 lower threshold
7 moving average
8 current sensor value

Upper override limit

For "Control OVERRIDE without alarm" and "Control OVERRIDE with alarm" methods. The path override is raised to control the sensor signal at the specified setpoint value and as a maximum up to this limit.

No alarm is generated when this limit is reached.
- Unit %
- Integer

Lower override limit

For "Control OVERRIDE without alarm" and "Control OVERRIDE with alarm" methods.

The path override is raised to control the sensor signal at the specified setpoint value and as a minimum up to this limit. When this limit is reached, an alarm is generated provided the "Control OVERRIDE with alarm" option has been selected.
- Unit %
- Integer

Setpoint value

For "Control OVERRIDE without alarm" and "Control OVERRIDE with alarm" methods. The path override is raised to control the sensor signal at the specified setpoint value and as a minimum up to this limit.

- Unit dependent on current sensor
- Floating point number

With the "Control OVR with alarm" method, an alarm is generated when this limit is reached.

i

Upper, lower threshold

Sliding average value is **less** than the switching point:

Upper threshold = sliding average + switching point x tolerance

Upper threshold = sliding average + switching point x tolerance

Sliding average is **greater** than the switching point:

Upper threshold = sliding average x (1 + tolerance)

Lower threshold = sliding average x (1 - tolerance)

i

Keyword "fixed limit"

Absolute fixed limit values are defined to a basic value around the monitoring signal. These limits can be used both for the fracture and also the overload. Fixed limits represent absolute tool protection.

i

Keyword "live threshold".

The monitoring limits are permanently determined and constantly carried from the processing signal and the tolerance value definition. For the evaluation, the change speed of the signal is used rather than its absolute level. The process is particularly suitable for detecting fast signal changes and therefore for the fracture method. By contrast, the process is excluded from the overload method (wearing diagram), since the absolute signal height is significant.

1.2.3.3 Create new methods

In order to create a new method, you may select a similar method from a set of templates and edit this as required or select an empty template and fill in the input fields in succession.

The procedure for filling in an empty template is described below.

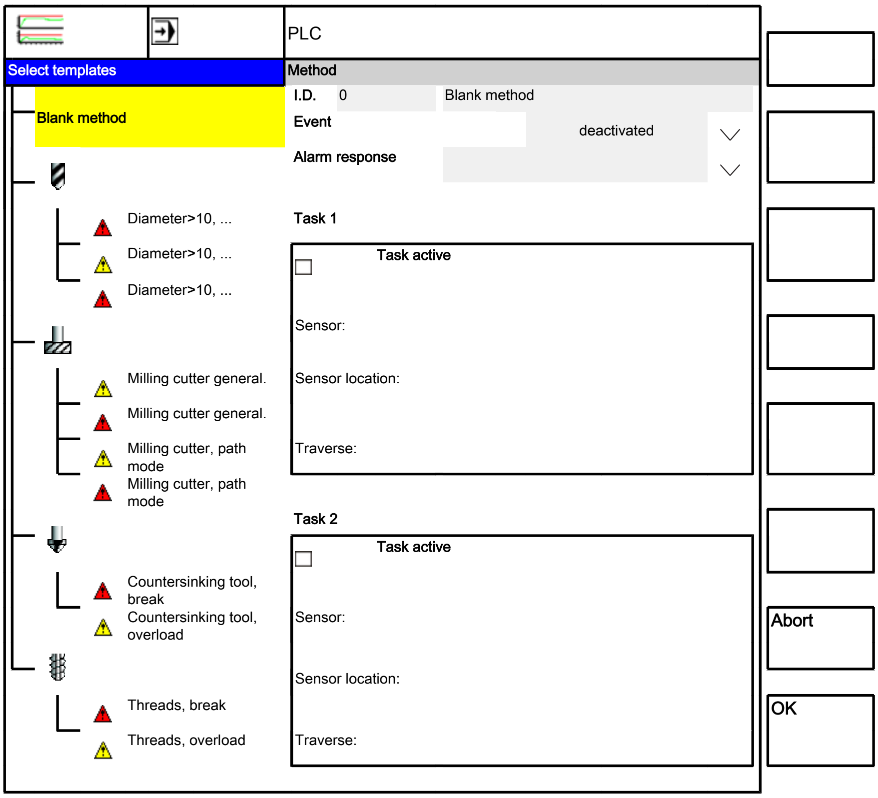
Precondition

☑ The "monitoring methods" window is opened.

New method

☒ Press the "New method" softkey.

↳ The "Select template" window is opened.



☒ Select the "empty methods" entry in the explorer structure of the left window section using the *Cursor keys*.

To use a template, select the entry required using the *Cursor keys*.

Ok

☒ Press the "Ok" softkey.

↳ The methods editor is opened.

Defining methods

Precondition

☑ The methods editor is opened and the cursor is in the "ID" input field.

TAB

☒ Press the *Tab* key.

↳ The suggested ID is accepted.
↳ The cursor moves to the next input field.

If necessary, you can change the ID using the alphanumeric keys.
The ID starts at 1000 and can be a maximum of 999999999.

K 1 L 2 ...

☒ Enter any method name using the *Alphanumeric keys*.

TAB

☒ Press the *Tab* key.

↳ The cursor moves to the "Event" input field.

☒ Press the *Insert* key.
↳ The selection menu containing the possible events opens.

☒ Select the required event using the *Cursor keys*.

☒ Press the *Enter* key.
↳ Enter the selected event and the corresponding alarm response.

Defining tasks

Task 1

☒ Press the "Task 1" softkey.
↳ The input window for defining task 1 is opened.

↳ The "Task active" checkbox is selected.

☒ Press the *Highlight* to activate the task.

↳ A tick appears in the checkbox.

Start repeat edit of the input fields.

TAB

☒ Press the *Tab* key.

↳ The next input field is selected.

☒ Press the *Insert* key.

↳ The selection menu opens.

☒ Select the required menu item using the *Cursor keys*.

☒ Press the *Enter* key.

↳ The selected menu item is accepted.

Re-edit all input fields.

A range of parameters will be displayed depending on the function you have selected.

Start to re-edit the parameters.

☒ Press the *Tab* key.

↳ The next input field is selected.

☒ Use the *Alphanumeric keys* to enter the parameter value.

Repeat parameter edit for all paramter fields.

Task 2

☒ Press the "Task 2" softkey to define a second task.

☒ Edit the input fields as described for task 1.

Ok

☒ Press the "Ok" softkey.

↳ The methods editor is closed and the new method is displayed in the "Monitoring methods" window.

1.2.3.4 Changing the method

An existing method is changed as follows:

Precondition

☑ The "monitoring methods" window is opened.

↑ ↓ ☒ Use the *Cursor keys* in the "Monitoring methods" window to select the methods you wish to change.

Change method ☒ Press the "Change method" softkey.

↳ The methods editor is opened.

For handling the method editor, see from Section "Defining methods" page 24

Ok ☒ Once you have made all changes, press the "Ok" softkey.

1.2.3.5 Copying method

To create similar monitoring methods, it makes sense to copy and change an existing method as required.

Precondition

☑ The "monitoring methods" window is opened.

↑ ↓ ☒ Use the *Cursor keys* in the "Monitoring methods" window to select the methods you wish to copy.

Copy method ☒ Press the "Copy method" softkey.

↳ The copied method is displayed and selected in the "Monitoring methods" window.
↳ The method name is prefixed with the symbol "~"

Change method ☒ Press the "Change method" softkey.

↳ The methods editor is opened.

For handling the method editor, see from Section "Defining methods" page 24

Ok ☒ Once you have made all changes, press the "Ok" softkey.

1.2.3.6 Delete method

Any monitoring methods that are no longer required can be deleted.

Precondition

☑ The "monitoring methods" window is opened.

↑ ↓

☒ Use the *Cursor keys* in the "Monitoring methods" window to select the methods you wish to delete.

Delete method

☒ Press the "Delete method" softkey.
↳ The monitoring method and associated tasks will be deleted.

There is the option of selecting several methods for deletion, see "Selecting methods" page 30

1.2.4 Importing/exporting methods

1.2.4.1 General

Import function

The import function can be employed to allow methods data from external databases to be used. The required data must be supplied via USB stick.
It is also possible to import data from machines with "Siemens 840D powerline" control.

Error handling during import operation:

- A syntax error in the import file will abort the import.
- If an error occurs during data transfer, the next file in line will be imported. An alarm message is also output.

Export function

Conversely, methods data can be provided by the IPM data base using the export function and loaded into databases on other machines.

Attention:
Data exported from machines with "Siemens 840D solution line" control can only be imported to machines with the same control.

Methods ID

Methods are stored in the database under a Methods ID. This ID is generated automatically when a new method is created. The system automatically increments the ID under which the method was stored in the database. User-specific methods have IDs from 1000 onwards.
A methods data record that already exists in the "IPM" menu is overwritten by an import procedure if the imported data record has the same methods ID.
When imported, a methods data record is automatically created and sorted if there was no previously existing data record with this methods identity.

⚠

> In order to avoid problems when transferring methods from one machine to another, methods must generally be produced on one machine only (master) from which they can then be exported. Failure to observe this instruction may result in existing methods on other machines being overwritten.
> Another alternative is defining unique Method IDs in the Methods Editor.

Import target

All imported data are stored in the IPM data base (definition data SGUD.DEF in SGUD area DEF.DIR).

Methods import of active tools

Methods data of tools that are already located in the magazine can be imported. The existing data record will be overwritten during the import operation.

Methods file

The methods files are stored as text files. The extension ***.mtd.** identifies the files.

minutes

A log showing any errors is created for each import or export operation.

USB interface

The USB interface is located on the main control panel, to the right on the operating panel for control functions.

Local drive

Import and export functions can also be used for the local drive. The data is stored in a specified directory on the local drive.

1.2.4.2 Export methods data

Precondition

☑ The "monitoring methods" window is opened.

Refer to
"Calling up the "Monitoring methods" window" **page 15**

Selecting methods

Select individual methods

- ☒ Use the *Cursor keys* in the "Monitoring methods" window to select the methods you wish to export.

Arbitrary selection of individual methods

- ☒ Press the *CTRL* key and hold down.
- ☒ Use the *Pointer (Mouse)* to point to the required method and select using the *left mouse key*.
- ☒ Select further methods using the *mouse*.
- ☒ Once you have selected all the methods you require, release the *CTRL* key.

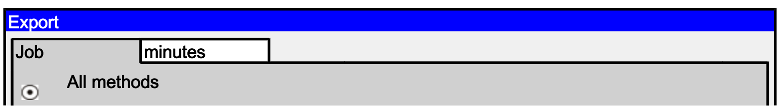
Selecting consecutive methods

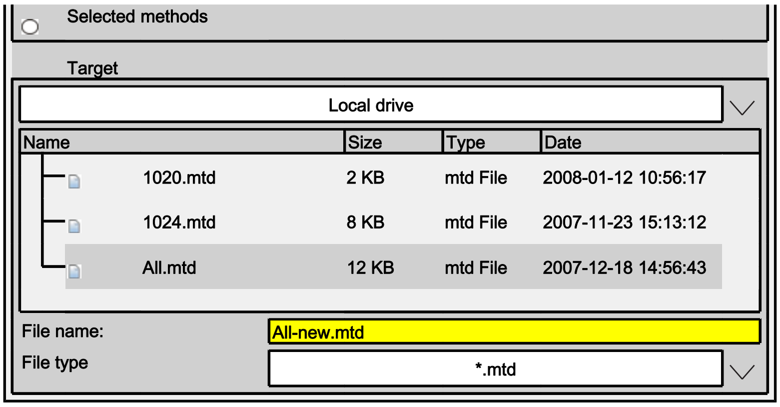
- ☒ Select the uppermost or lowermost methods you require using the *Cursor keys*.
- ☒ Press the *Shift* key and hold down.
- ☒ Select further required methods using the *Cursor keys*.
- ☒ Once you have selected all the methods you require, release the *Shift* key.

Exporting methods

- ☒ Press the "Export" softkey.

↳ The "Export" window, "Order" tab is opened.





TAB

☒ Press the *Tab* key to select which methods are to be exported.

- All methods: All methods displayed in the "Monitoring methods" window.
- Selected methods: The methods selected in the "Monitoring methods" window.

☒ Press the *Insert* key.

↳ The methods selection is confirmed.

TAB

☒ Press the *Tab* key until the cursor is in the input field for the drive selection.

☒ Press the *Insert* key.
↳ The selection menu containing the possible drives opens.

☒ Select the local drive using the *Cursor keys*.

☒ Press the *Enter* key.

↳ The selected drive is entered.
↳ The contents of the drive are displayed.

TAB

☒ Press *Tab* to select the "File name" input field.

K 1 L 2 ...

☒ Use the *Alphanumeric keys* to enter the file name under which the exported methods are to be saved to the data carrier.

?

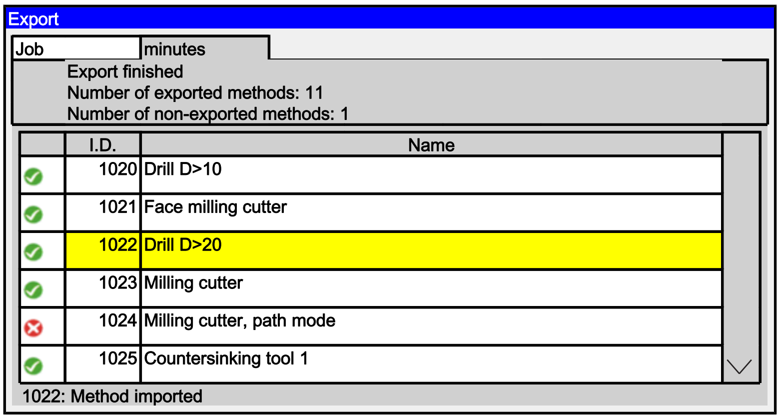
Entered incorrect data? You wish to abort the export operation.

☒ Confirm by pressing the "Escape" softkey to close the "Export" window and abort the operation.

Ok

☒ Press the "Ok" softkey.

↳ The data export is started.
↳ A file with an .mtd extension is created, in which all exported methods are contained.
↳ Once the data export operation is complete, the "Export" window and "Log" tab are opened.



↳ The "Log" tab shows which methods have been successfully exported:

	green	Successfully imported
	red	Not imported

Ok

☒ Press the "Ok" softkey again to complete the export operation.

1.2.4.3 Importing methods data

Precondition

☑ The file to be imported is on the data carrier.

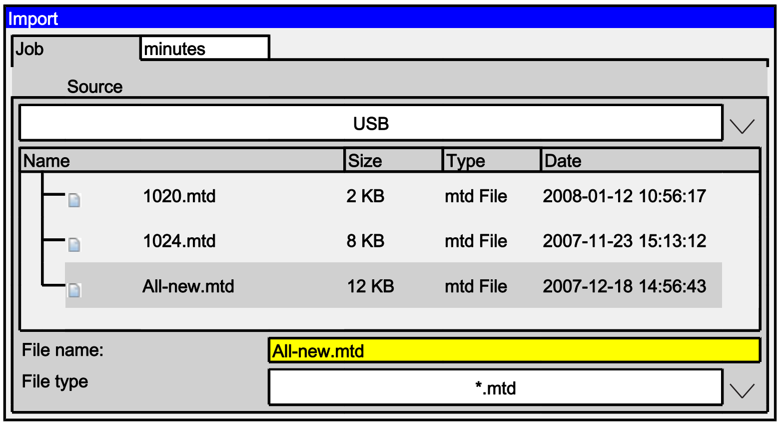
☑ The "monitoring methods" window is opened.

Refer to
"Calling up the "Monitoring methods" window" **page 15**

Import

☒ Press the "Import" softkey.

↳ The "Import" window, "Order" tab is opened.



Actual display may differ.

☒ Press the *Tab* key until the cursor is in the input field for the drive selection.

☒ Press the *Insert* key.
↳ The selection menu containing the possible drives opens.

☒ Use the *Cursor keys* to select the drive containing the file to be imported.

☒ Press the *Enter* key.

↳ The selected drive is entered.
↳ The contents of the drive are displayed.

☒ Use the *Cursor keys* to select the file that is to be imported into the IPM database.
↳ The file name is accepted into the "File name" field.

?

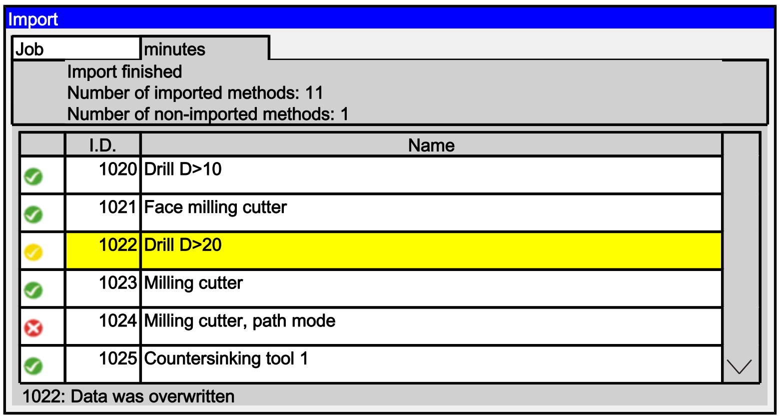
Entered incorrect data? You wish to abort the import operation.

☒ Confirm by pressing the "Escape" softkey to close the "Import" window and abort the operation.

Ok

☒ Press the "Ok" softkey.

↳ The import operation is started.
↳ Once the data import operation is complete, the "Import" window and "Log" tab are opened.



↳ The "Log" tab shows which methods have been successfully exported:

- green Successfully imported
- yellow Successfully imported, existing methods overwritten.
- red Not imported

Ok

☒ Press the "Ok" softkey again to complete the import operation.

1.2.5 Assigning monitoring methods to a tool

1.2.5.1 General

Input window

In a subsequent step, the specified monitoring methods are assigned to the real tools of the machine. The monitoring methods in the "Tool wear" list can be assigned.

The "tool wear" list is available from the "Parameters" menu.

User-specific methods (> 1000)

A method is assigned by entering the corresponding method ID. This means, the user-specific methods in the address range > 1000 can be assigned.

Assignment per tool

In the "Tool wear" list, the monitoring methods must be individually assigned for each tool.
A maximum of two monitoring methods can be assigned to each tool.

HELLER

1.2.5.2 Assigning monitoring methods

☑ Corresponding monitoring methods are created in the "IPM" area.

☑ The tool is created with all tool data.

See Operator Manual, "Tool management system", 'Tool data' menu" section.

Opening the "Tool wear" list

☒ Press *Data menu key* to call-up the machine's basic menu.

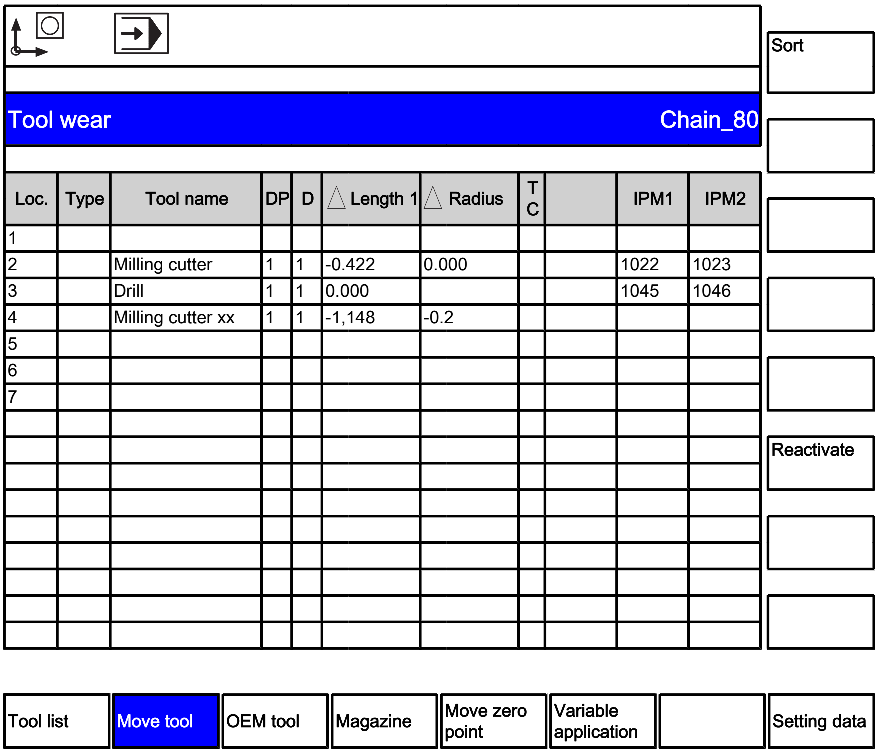
Parameter

☒ Press the "Parameter" softkey.
↳ The "Tool" area is opened.

Tool wear

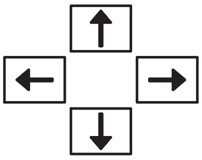
☒ Press the "Tool wear" softkey.

↳ The "Tool wear list" is opened.

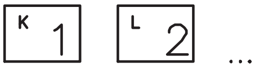


Loc.	Type	Tool name	DP	D	△ Length 1	△ Radius	T C		IPM1	IPM2
1										
2		Milling cutter	1	1	-0.422	0.000			1022	1023
3		Drill	1	1	0.000				1045	1046
4		Milling cutter xx	1	1	-1,148	-0.2				
5										
6										
7										

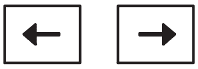
Actual display may differ.



☒ Use the *Cursor keys* to select the "IPM1" column of the tool to which a monitoring method is to be assigned.



☒ Use the *Alphanumeric keys* to enter the methods ID of the required monitoring method.



☒ Press the *Cursor keys* to select column "IPM2" and assign a further monitoring method if necessary.

1.2.6 Programming the integrated process monitoring system

1.2.6.1 Syntax of programming instructions

Overview

System cycles, language commands

A series of system cycles (CS_IPM..) are available that enables Integrated Process Monitoring (IPM) to be controlled from the NC program.
Once the monitoring method, which becomes effective with the tool change cycle, has been called up, your data can be changed, if necessary, through special language commands (CS_IPM...) so that they specifically matches the machining process. These block-related changes do not affect the tool-related data which have been specified during tool assignment.
Block-related data is rejected after the next tool change and replaced by the tool-related data.

CS_IPM....

The following system cycles and corresponding language commands are available:

System cycle	Explanation
`CS_IPMON( )`	Enable process monitoring
`CS_IPMOF( )`	Disable process monitoring.
`CS_IPMMW( )`	Write current monitoring data.
`CS_IPMMR( )`	Read current monitoring data.
`CS_IPMCO( )`	Command to IPM.

CS_IPMVR( )	Read min./max. values from monitoring data.
CS_IPMMS( )	Select monitoring method.
CS_ESCLAB( )	Write alternative label.
CS_ESCON( )	Activate alternative strategy.
CS_D( )	Select cutting tip.
CS_IPMWL( )	Write (read) min./max. values to (from) file.
CS_IPMTO( )	Set spindle tool orientation.

CS_IPMON(...)

Explanation Enable tool monitoring

Syntax CS_IPMON (STRING,VALUE)

Input/activation

STRING	" "	Enable all monitoring methods together.
	"BREAK"	Enable break method.
	"OVERLOAD"	Enable overload method.
VALUE	1 or 2	Optional data for method group. If the parameter is omitted, all method groups are addressed.

Examples

CS_IPMON	Enable all monitoring methods in Method Group 1.
CS_IPMON( )	Enable all monitoring methods in Method Group 1.
CS_IPMON("")	Enable all monitoring methods in Method Group 1.
CS_IPMON("BREAK")	Enable break method in Method Group 1.
CS_IPMON("OVERLOAD")	Enable overload method in Method Group 1.
CS_IPMON(,2)	Enable all methods in Method Group 2.
CS_IPMON("OVERLOAD",2)	Enable overload method in Method Group 2.

If the STRING is omitted: all monitoring methods are to be enabled

If the VALUE is omitted, all method groups are addressed.

CS_IPMOF(...)

Explanation Enable tool monitoring

Syntax CS_IPMOF (STRING,VALUE)

Input/activation

STRING	" "	Disable all monitoring methods together.
	"BREAK"	Disable break method.
	"OVERLOAD"	Disable overload method.
VALUE	1 or 2	Optional data for method group. If the parameter is omitted, all method groups are addressed.

Examples

`CS_IPMOF`	Disable all monitoring methods in Method Group 1.
`CS_IPMOF( )`	Disable all monitoring methods in Method Group 1.
`CS_IPMOF("")`	Disable all monitoring methods in Method Group 1.
`CS_IPMOF("BREAK")`	Disable break method in Method Group 1.
`CS_IPMOF("OVERLOAD")`	Disable overload methods in Method Group 1.
`CS_IPMOF(,2)`	Disable all methods in Method Group 2.
`CS_IPMOF("OVERLOAD",2)`	Disable overload methods in Method Group 2.

If the STRING is omitted: all monitoring methods are to be disabled

If the VALUE is omitted, all method groups are addressed.

CS_IPMMW(...)

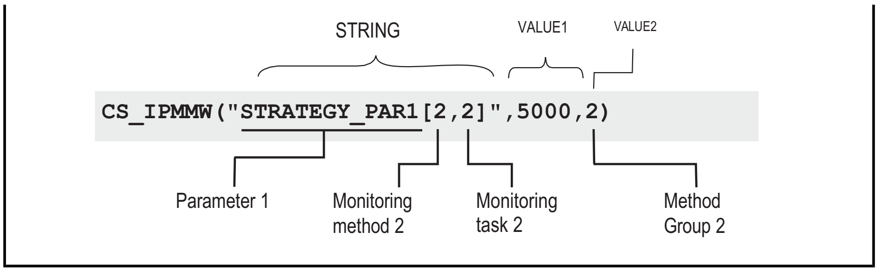
Parameters in general

Explanation Set parameters for monitoring methods (current value).

Syntax CS_IPMMW(STRING,VALUE1,VALUE2)

Input/activation

STRING	Parameter name as string variable (GUD name excluding IPM_METHOD) Example: "Upper limit" parameter setting.
VALUE1	Parameter value (REAL variable type)
VALUE2	Optional data for Method Group 2



Parameters in details, examples

Changes to the following "STRATEGY_PARx[...]" parameters are effective immediately.

Parameter 1

CS_IPMMW("STRATEGY_PAR1[2.2]",5000)
Set parameter 1 of the monitoring process for method group 1, method 2, task 2 to a value of 5000.

Meaning of Parameter 1:	
With "fixed limit" process:	Upper limit
With "moving threshold" process:	Tolerance

Parameter 2

CS_IPMMW("STRATEGY_PAR2[1.2]",777)
Set parameter 2 of the monitoring process for method group 1, method 1, task 2 to a value of 777.

Meaning of Parameter 2:	
With "fixed limit" process:	Lower limit
With "moving threshold" process:	Switching point

Parameter 3

CS_IPMMW("STRATEGY_PAR3[2.1]",711)
Set parameter 3 of the monitoring process for method group 1, method 2, task 1 to a value of 711.

Meaning of Parameter 3:	
With "fixed limit" process:	Average factor for sensor
With "moving threshold" process:	Average factor for threshold

Parameter 4

CS_IPMMW("STRATEGY_PAR4[1.1]",15)
Set parameter 4 of the monitoring process for method group 1, method 1, task 1 to a value of 15.

Meaning of Parameter 4:	
With "fixed limit" process:	Missing and/or blunt limit
With "moving threshold" process:	Average factor for sensor

Parameter 1

CS_IPMMW("STRATEGY_PAR1[1.2]",995,2)

Set parameter 1 of the monitoring process for method group 2, method 1, task 2 to a value of 995.

Changes to the other parameters only become effective when a new "CS_IPMMON" instruction is given.

This should however only be done in exceptional cases, since the significance of the strategy parameters could be changed as a result.

Event

CS_IPMMW("EVENT[1]",1)
Set monitoring event for method group 1, method 1 to "break".

Target

CS_IPMMW("JOB_MASK[1]",1)
Only 1 task is present in Methods Group 1, Method 1.

Connection

CS_IPMMW("JOB_CONNECT[1]",1)
The tasks of Method Group 1, Method 1 are connected as "Or".

Alarm response

CS_IPMMW("ALARM_RESPONSE[2]",1)
The response of method group 1, method 2 should cause an alarm to be displayed.

Sensor, sensor location

CS_IPMMW("SENSOR_ID[1.2]",257)
The signal for the monitoring of Method Group 1, Method 1, Task 2 should be the setpoint torque value of the 1st machine axis.

Process

CS_IPMMW("STRATEGY[2.1]",1.2)
The monitoring process of Method Group 2, Method 2, Task 1 should be "current value, fixed limit".

Determine the basic mean value factor for the "current value, live threshold" process

current value, live threshold

With the "current value, moving threshold" process, certain tools or spindles encounter the problem of the basic signal being "noisy" or "unsteady".

Strategy parameter 4

Another process "sliding average/moving threshold" solves this problem by pre-filtering the signal. The filtering operation is handled by another 4th parameter, which acts as a variable for noise suppression:

- Average factor for sensor (IPM window)
- Strategy parameter 4 (programming)

Strategy parameter 4 can be used through the language command "CC_IPM_METHOD_WR" and can be changed "online".

Example

CC_IPM_METHOD_WR("STRATEGY_PAR4[1.1]",2)
Strategy parameter 4 of Method 1, Task 1 is set to 2.

Basic mean factor

Recommendation:
The signal to be processed is monitored initially with the "sliding average, fixed limit" process and steadied with the aid of the average factor for "threshold". The determined average factor for "sensor" is then used as the average factor for "threshold" (basic average factor) with the "current value, moving threshold" process.

CS_IPMMR(...)

Explanation

Read parameters for monitoring methods (current value).

Syntax

CS_IPMMR (RET_VALUE,STRING,VALUE)

Input/activation

RET_VALUE	Return value of current value (REAL variable type)
STRING	Parameter name as string (GUD name excluding IPM_METHOD). Example: "EVENT" for monitoring event.
VALUE	Optional data for Method Group 2.

Example

CS_IPMMR(R0"STRATEGY_PAR2[1.2]")
Read out Parameter 2 of the monitoring process for Method Group 1, Method 1, Task 2 in the RO variable.

CS_IPMMR(R1"STRATEGY_PAR2[2.1]",2)
Read out Parameter 2 of the monitoring process for Method Group 2, Method 2, Task 1 into the R1 variable.

CS_IPMCO(...)

Explanation

Execute IPM command

Syntax

CS_IPMCO (STRING,VALUE1, VALUE2)

Input/activation

String	Value range
`"CLEARTOWVAL"`	Delete all min/max values. VALUE1: No meaning VALUE2: Definition method group 2
`"CLEARTOWVAL_BREAK"`	Delete min./max. values of break monitoring. VALUE1: No meaning VALUE2: Definition method group 2

String	Value range
"CLEARTOWVAL_OVERLOAD"	Delete min./max. values of overload monitoring. VALUE1: No meaning VALUE2: Definition method group 2
"NEXT_NOT_EXECUTABLE"	Set next block to "not executable". Instruction required if halting at end of block is not possible. The CC command is not effective until the next executable block. VALUE1, VALUE2: No meaning
"NEXT_EXECUTABLE"	Set next block to "executable". The CC instruction is effective immediately. VALUE1, VALUE2: No meaning
"CLEARALARM_BREAK"	Reset all "break" alarms. VALUE1, VALUE2: No meaning
"CLEARALARM_OVERLOAD"	Reset all "overload" alarms. VALUE1, VALUE2: No meaning
"CLEARALARM"	Reset all alarms. VALUE1, VALUE2: No meaning
"LOWLIMIT_DELAYTIME"	Activate once only cutting monitoring. VALUE1: Time in seconds. VALUE2: No meaning.
"LOWLIMIT_DELAYTIME_REPEAT"	Activate repeatable cutting monitoring. VALUE1: Time in seconds. VALUE2: No meaning.
"LOWLIMIT_DELAYDIST"	Delete all min/max values. VALUE1: No meaning VALUE2: Definition method group 2
"LOWLIMIT_DELAYDIST_REPEAT"	Activate repeatable cutting monitoring. VALUE1: Path distance in mm. VALUE2: No meaning
"ALARM_OFF"	Suppress alarm output. VALUE1, VALUE2: No meaning
"ALARM_ON"	Enable alarm output. VALUE1, VALUE2: No meaning
"SIGNAL_FACTOR"	Signals of an axis (x=1..31) to be evaluated using a factor. VALUE1: Evaluation factor. VALUE2: No meaning.
"RESET_METHODGROUPS"	Set methods of all method groups as ineffective. VALUE1: No meaning. VALUE2: No meaning.
"CONFIGURE_METHODGROUPS"	Execute complete configuration of all methods/~-groups. VALUE1: No meaning. VALUE2: No meaning.
"CHECK_WEAR"	"Blunt" monitoring: With overload methods combined with strategy parameter 4 (not with "moving threshold" process). The programmed parameter 4 is compared with the current block average. Blunt alarm generated if the block average is greater than parameter 4. VALUE1: No meaning. VALUE2: No meaning.

String	Value range
"CHECK_MISSING"	"Missing" monitoring: With break methods combined with strategy parameter 4 (not with "moving threshold" process). The programmed parameter 4 is compared with the current block average. Missing alarm generated if the block average is smaller than parameter 4. VALUE1: No meaning. VALUE2: No meaning.
"CHECK_WEAR_AND_MISSING"	"Blunt" and "missing" check simultaneously. VALUE1: No meaning. VALUE2: No meaning.
"CLEAR_BLOCK_AVERAGEVAL"	Reset current block average for all monitoring tasks. VALUE1: No meaning. VALUE2: No meaning.
"UPPER_OVERRIDE_CHECK"	Disable blunt check when the upper override value is exceeded. Example: With VALUE1 = 110 and an override value of 115, for example, the alarm generation for blunt is prevented. Appropriate value range: 100 ... 200. VALUE2: No meaning.
""LOWER_OVERRIDE_CHECK"	Disable cutting and missing check when the lower override value is not reached. Example: For VALUE1 = 70, alarm generation for cutting and missing check is inhinged at an override value from 60, for example. Appropriate value range: 0 ... 100. VALUE2: No meaning.
"SET_EVENT_STATE"	Enables the ON/OFF statuses of individual or all events to be controlled. (from IPM1.1/13). The cc_ipm_value_rd("EVENT_STATE") command is useful in this respect. For example, if you want to temporarily disable monitoring during a measuring cycle, this the original status can be restored using this command. The value to be written is bit-coded: Bit 0 = log.1: all monitoring runs are active (remaining bits are then irrelevant). Bit 1 = log 1: break monitoring is active, Bit 2 = log 1: overload monitoring is active
"FEEDRATE_AVERAGE_FACTOR"	Enables input signal for the "controlled override" method to be pre-filtered. (from IPM1.2/9). This value defaults to 10 (after NC reset). VALUE2: No meaning.
"FEEDRATE_AIR_CUT_LIMIT"	Sets limit for air cutting detection with "controlled override" method (from IPM1.2/9). If the input signal falls below this limit, the path override is set to the value defined by the "FEEDRATE_AIR_CUT_FEED_FACTOR" parameter. This value defaults to 0 (after NC reset), i.e. air cutting detection is disabled. VALUE2: No meaning.
"FEEDRATE_AIR_CUT_FEED_FACTOR"	Sets value for path override during air cutting detection if "controlled override" method is selected (from IPM1.2/9). If the input signal falls below the "FEEDRATE_AIR_CUT_LIMIT" value, the path override is permanently set to the value defined by the "FEEDRATE_AIR_CUT_FEED_FACTOR" parameter. This value defaults to 0 (after NC reset), i.e. air cutting detection is disabled. VALUE2: No meaning.

String	Value range
"FEEDRATE_GAIN"	Sets the P-component of the (PI) controller if the "controlled override" method is selected. (from IPM1.2/9). If the value is 0, which is also the default (after NC reset), this is a pure I-controller with integration time "FEEDRATE_TIME". This (default) value has been optimised and should not be changed. VALUE2: No meaning.
"FEEDRATE_TIME"	Sets the I-component of the (I or PI) controller if the "controlled override" method is selected. (from IPM1.2/9). Following initialisation (after NC-reset), an I-controller with time constant 4s is active. If a PI controller is used, the P-component should be adjusted using "FEEDRATE_GAIN". This value (default) has been optimised and should not be changed. VALUE2: No meaning.
"FEEDRATE_UP_FACTOR"	This (default) value has been optimised and should not be changed. VALUE2: No meaning.
"FEEDRATE_DOWN_FACTOR"	This (default) value has been optimised and should not be changed. VALUE2: No meaning.
VALUE1	Parameter value (REAL variable type) is only relevant at "LOWLIMIT_...", "SIGNAL_FACTOR..", "..OVERRIDE_CHECK" , "SET_EVENT_STATE" and "FEEDRATE_.." Irrelevant
WERT2 (1bzw.2)	Optional data for method group. Relevant only for "CLEARTOWVALUE..." and "SET_EVENT_STATE". If the parameter is omitted, all method groups are addressed.

Examples

CS_IPMCO("CLEARTOWVAL", ,2)
Delete all tow indicator values from the monitoring data of method group 2.

CS_IPMCO("LOWLIMIT_DELAYDIST", 10)
Activate cutting check after a path distance of 10 mm.

CS_IPMCO("CHECK_WEAR_AND_MISSING")
Trigger "blunt" and "missing" check.

Cutting monitoring

General

The cutting check (from IPM V1. 1/5) enables a lower limit of a break method to be monitored. This ensures a missing or broken tool is detected

Syntax

CS_IPMCO("LOWLIMIT_DELAYTIME",x)

CS_IPMCO("LOWLIMIT_DELAYDIST",y)

x = time value in seconds
Y = distance in mm
After x seconds or y mm path distance (TCP = tool centre point), the current or determined value is checked against the lower limit respectively. If this value is below the limit, an alarm is triggered. and the lower limit is then no longer checked.

Syntax

CS_IPMCO("LOWLIMIT_DELAYTIME_REPEAT",x)

CS_IPMCO("LOWLIMIT_DELAYDIST_REPEAT",y)

x = time value in seconds
y = distance in mm
After x seconds or y mm path distance, the current and/or or averaged value is checked against the lower limit. If this value is below the limit, an alarm is triggered. The same process is then repeated, if the current or averaged value falls below the lower limit.

Notes

The cutting check functions only with "break" methods. When a G0 block is reached, the counted path distance is reset. The measurement of x and/or y only begins again during the next feed block. TCP means "Tool centre point".
When programming a distance in mm, please note that activation of a tool radius offset (e.g. G41) relates to the path of the tool centre point and not to the path on the contour.
From IPM V1.1/7: Cycle CS_IPMCO("CHECK_LOWER_OVERRIDE",x) can be used to prevent a cutting alarm being triggered if a feed override x is not reached.

Blunt and missing monitoring

General

Blunt and missing monitoring (from IPM V1. 1/7) enables a "surface evaluation" to be effected using a machining section to be defined.

Syntax

CS_IPMCO("CHECK_WEAR")

CS_IPMCO("CHECK_MISSING")

CS_IPMCO("CHECK_WEAR_AND_MISSING")

General

At precisely this point in time, all tasks defined by the "averaged value, fixed limit and blunt/missing limit" process are compared with a value present in strategy parameter 4 (or alternatively in the Method Editor in the "missing or blunt limit" parameter) and the current block value.

For fracture methods, the "missing" alarm is triggered if: current block average value < strategy parameter 4.
For overload methods, the "blunt" alarm is triggered if: current block average value > strategy parameter 4.
The block average can be read or deleted at any time. The block average is created for all processes (including "current value, moving threshold"). With moving threshold, strategy parameter 4 is reserved to create the block average and therefore unavailable for missing/blunt monitoring.
The following language commands are available to read/delete the block average:
CS_IPMVR(R0,"BLOCK_AVERAGEVAL[1.2]",1)
Meaning: The block average value for method group 1, method 1, task 2 is read into the calculation parameter r0.
CS_IPMCO("CLEAR_BLOCK_AVERAGEVAL")
Meaning: All block average values for all tasks are deleted. The block average is then recreated.

From IPM V1.1/7:
CS_IPMCO("CHECK_LOWER_OVERRIDE",x)
A missing alarm can be prevented from being triggered if a feed override x is not reached
CS_IPMCO("CHECK_UPPER_OVERRIDE",x)
a blunt alarm can be prevented from being triggered if a feed override x is exceeded.
With blunt and missing monitoring, the task links are not considered (behaviour as with an 'OR' link). The check (CS_IPMCO) can also be performed whilst rapid traverse G0 is active.

CS_IPMVR(...)

Explanation

Read IPM values.

Syntax

CS_IPMVR (RET_VALUE,STRING,VALUE)

Input/activation

RET_VALUE	Return value of current value (REAL variable type)
STRING	The following values to be read as STRING: "UPPER_TOWVAL": read upper tow value. - "LOWER_TOWVAL": read lower tow value - "BLOCK_AVERAGEVAL": read block average. - "EVENT_STATE": read event state. - "DEBUGVAL": read debug value (for service only).
VALUE	Optional data for method group. Irrelevant for "DEBUGVAL". If the parameter is omitted, method group 1 is addressed.

Example

CS_IPMVR(R0,"UPPER_TOWVAL[1.2]")

Save upper tow value for Method Group1, Method 1, Task 2 to the R0 variable

CS_IPMVR(R1,"UPPER_TOWVAL[1.2]",2)
Save tow value for Method Group2, Method 1, Task 2 to the R1 variable

CS_IPMVR(R2,"BLOCK_AVERAGE[2.1]",2)
Save block average for Method Group2, Method 1, Task 2 to the R2 variable

CS_IPMVR(R3,"EVENT_STATE",1)
Store event states of method group 1 (from IPM1.1/13) in Variable R3. Reads the ON/OFF statuses of certain or all events. The CS_IPMCO("SET_EVENT_STATE"..) command is useful in this respect.
For example, if you want to temporarily disable monitoring during a measuring cycle, the previous status can be restored using this command.
The read result is bit-coded:

Bit 0 = log.1:	all monitoring operations are active (The remaining bits are then irrelevant)
Bit 1 = log 1:	Break monitoring is active
Bit 2 = log 1:	Overload monitoring is active

CS_IPMVR(R4,"DEBUGVAL[1]")
Read debug value 1. Store result in variable R4. This function is intended for diagnostic purposes.

CS_IPMMS(...)

Explanation

Method selection.
This cycle is not usually required, since it is automatically selected along with a tool change or cutting tip change (CS_D). Theoretically, however, it is possible to access alternative methods (which have to be loaded) at any time.

Syntax

CS_IPMMS (VALUE1,VALUE2,VALUE3,STRING)

Input/activation

VALUE1	Method key of Method No. 1.
VALUE2	Method key of Method No. 2.
VALUE3	Optional data for method group. If the parameter is omitted, all method groups are addressed.
STRING	Optional data of a tool identifier (for alarm messages).

Example

CS_IPMMS(1)
As Method 1, the method is activated in Method Group 1 (default) using method key 1.

CS_IPMMS(3,4,2)
The method with methods key 3 is activated as method 1 and the method with methods key 4 in method group 2 is activated as method 2.

CS_IPMMS(,6,1)
As Method 2, the method is activated in Method Group 1 using method key 6.

CS_D(...)

Explanation

Select cutting tip with using additional method selection.
The cycle must be used, since the standard tip selection (e.g. D2 and/or D=2) does not automatically activate the method allocated to the cutting tips.

Syntax

CS_D (VALUE)

Input/activation

VALUE	Number of cutting tip (integer).

Example

CS_D(3)
Select cutting tip no. 3 (corresponds to D3 and/or D=3).

CS_IPMWL(...)

Explanation

Write/read and/or learning/monitor tow values to a file.

Syntax

CS_IPMWL (STRING1,STRING2,STRING3)

Input/activation

STRING1	Possible modes for which the cycle is to be activated: - "LEARNING" - "MONITORING" - "LOGFILE" - "COMMAND"

STRING2	With STRING1="LEARNING", "MONITORING", the current mode is possible ("LEARNING", "MONITORING"). For STRING1="LOGFILE": Name of logfile (optional) [1] For STRING1="COMMAND": command type possible. - "CLEARLOGFILES": clear all log files. - "START_LEARNING_MONITORING": start learning/monitoring. - "ALL_ON": switch-on all logs - "ALL_OFF": switch-off all logs - "LOWERTOW_ON": lower tow value on - "LOWERTOW_OFF": lower tow value off - "UPPERTOW_ON": upper tow value on - "UPPERTOW_OFF": Upper tow value from - "BLOCKAVERAGE_ON": block average on (from IPM V1.1/7). - "BLOCKAVERAGE_OFF": Block average value off. - "TEXTFORMAT_ON": Text format on. - "TEXTFORMAT_OFF": Text format off (Excel table) [2]. - "HEADERLOGFILELINE_ON": show header line . - "HEADERLOGFILELINE_OFF": hide header line. - "LASTLOGFILELINE_ON": show last line. - "LASTLOGFILELINE_OFF": hide last line. - "SETMONITORINGCOUNTER": set monitoring counter (with cross-access).
STRING3	For STRING1="LOGFILE": text that can appear as the header (optional). For STRING1="LEARNING","MONITORING": name of learn file (optional). For STRING1="COMMAND" and STRING2=" SETMONITORINGCOUNTER", located here is the "counter value" for cross-entry into the program as a string, e.g. "3" for the 3rd phase or tool.

1) If the file is full, it is deleted and re-written.

2) One line, data separated by commas

Programming example

```
DEF STRING[20] LV_ACTMODE="MONITORING".
;Default setting for reporting.
CS_IPMWL("COMMAND","ALL_ON")
;delete logfiles.
CS_IPMWL("COMMAND","CLEARLOGFILES")
;start from learning/monitoring.
CS_IPMWL("COMMAND","START_LEARNING_MONITORING")
CS_TOOL("1")
;the learned limits are read only in "MONITORING" mode.
CS_IPMWL("Monitoring", LV_ACTMODE)
;The machining program for tool T1 is located here.
;the current upper limits are written only in "LEARNING" mode.
CS_IPMWL("LEARNING", LV_ACTMODE)
;Store current tow indicator values in the log file.
```

```
CS_IPMWL("LOGFILE", , "Programmstart:"<<$A_Hour<<":"<<
$A_MINUTE<<":<<$A_SECOND)

CS_TOOL("2")
;the learned upper limits are read only in "MONITORING" mode.
CS_IPMWL("MONITORING", LV_ACTMODE)
;The machining program for tool T2 is located here.
;the current upper limits are written only in "LEARNING" mode.
CS_IPMWL("LEARNING", LV_ACTMODE)
;Store current tow indicator values in the log file.
CS_IPMWL("LOGFILE")
```

In the above example, the upper limits of the monitoring methods can be "learned"=="LEARNING" and "monitored"=="MONITORING".

The "LEARNING" mode can be activated by initialising the variable LV_ACTMODE="LEARNING" before the program is started. After a learning run, monitoring can take place by writing to the variable LV_ACTMODE="MONITORING".

The leaned values stored during the learning operation are saved to a file in the current directory (e.g. workpiece directory) (default: IPMLEARN_CHANx.MPF).

With overload methods, the limits are set to 120% of the learned peak values and with break methods to 150% of the learned peak values.

The cycle also enables a log file to be generated. At arbitrary places in the machining program, the lower and upper limits and the block average can be logged in the log files (default: IPMLOG_CHANx_y.MPF), which are also stored in the current directory, at any point in the machining program. Comment: x=channel number, y=file number.

CS_IPMTO(...)

Explanation

Set spindle tool orientation.
This tool orientation remains valid until the next cutting tip and/or tool change. The orientation vector can be any length. The vector (0,0,0) deletes the predefined orientation again.

Syntax

CS_IPMTO (VALUE1,VALUE2,VALUE3)

Input/activation

VALUE1	Geoaxis 1 and/or X-component of the spindle vector.
VALUE2	Geoaxis 2 and/or Y-component of the spindle vector.
VALUE2	Geoaxis 3 and/or Z-component of the spindle vector.

Example

CS_IPMTO(0,TAN(30),1)

Tool that points 30 degrees upwards relative to the horizontal.

CS_IPMTO(1,0,0)
Tool that points towards the X-axis (e.g. angular head).

CS_IPMTO(0,0,0)
Delete tool orientation.

1.2.6.2 Controlling monitoring via synchronous actions

$AC_MARKER

$AC_MARKER[8] is available for synchronous actions. This is bitcoded. The individual bits mean the following:

Bit 0=1	Routed deactivation of all monitoring events (break, overload).
Bit 1=1	Routed deactivation of the break event.
Bit 2=1	Routed deactivation of overload monitoring.

Example

```
N10 G0 G53 G40
N20 X0
N30 $AC_MARKER[8]=1; Deactivate all events
N40 G1 F5000
N50 when $AA_IM[MA_X]>=100 DO $AC_MARKER[8]=0
N60 when $AA_IM[MA_X]>=200 DO $AC_MARKER[8]=1
N70 X30
```

In the NC block N70, monitoring is enabled for as long as the X-axis (corresponds to machine axis MA_X) is located between positions 100 and 200.

1.2.6.3 Alternative Strategy (option)

General

System cycles

The NC programmer has the option of switching to alternative program in his NC program such an event occur (IPM, BTD and/or BBK or EWS). User-friendly and understandable system cycles are available for this purpose.

Programming rules

Certain programming rules must be observed:

- The alternative strategy can only be programmed at the 1st programming level.
- The first executable lines in the program must be the cycle call up `CS_ESCON(bit bar)`. The bit bar can be used to select the faults that will trigger the alternative strategy. Bit1 -> Integrated Process Monitoring IPM.
- The alternative destination to be reached in the event of a fault must be programmed by the NC programmer in the program sequence using `CS_ESCLAB("Label")`.
- The "Label" alternative destination must be programmed at the 1st programming level.
- Integrated Process Monitoring is enabled via `CS_IPMON("Mode")`. The monitoring type is selected through mode. (only with IPM).
- Integrated Process Monitoring is disabled through `CS_IPMOF("Mode")` in line with the mode. (only with IPM).
- Alternative strategy is automatically deselected at the end of the program.

System cycles

CS_ESCON(...)

Explanation

System cycle "CS_ESCON" is used to anchor the alternative strategy in the NC program.
The system cycle must always be at the start of the program. The alternative strategy is disabled at the end of the program.

Syntax

CS_ESCON (BIT BAR)

Input/activation

BIT 0:	1	No usable tool activated.
BIT 1:	1	Integrated Process Monitoring IPM activated. (Break method)
BIT 2:	1	Tool break monitoring activated

Example

BIT BAR	BIT 2	Bit 1	Bit 0	
	1	0	0	= 4

`CS_ESCON(4)`	Activate tool break monitoring
``CS_ESCON(`B100`)``	Activate tool break monitoring
``CS_ESCON(`B10`)``	Activate IPM (break method)

CS_ESCLAB(...)

Explanation

The "CS_ESCLAB" is used to declare the alternative destination (label) to which the program should skip should a monitoring event occur. The alternative target must be positioned before each program section.
If no "IPM" option is installed, a main run stop is required to describe the label. Otherwise, the label is written through a NCK-OEM function at the main run time point. This is realised in the system cycle, so that the programmer can always call up the same cycle, irrespective of the IPM option.

Syntax

CS_ESCON ("PARAM")

Input/activation

PARAM	String variable.

CS_ESCEV(...)

Explanation

You do not have to program the "CS_ESCEV" system cycle. The cycle is automatically selected in the event of a fault.
This triggers the following actions:

- A M64 is output, in order to set the tool to the "reject" status (NOK).
- The cycle is used as a branch address distributor for alternative labels when the main program is restarted after a fault.

1.2.6.4 Programming example

`%_N_AWS_TEST1_MPF`	
`;$PATH=/_N_WKS_DIR/_N_NCTEST_WPD`	
`N10 CS_ESCON(2)`	Activate Bit 1 tool monitoring.
`N20 CS_ESCLAB("AUSW");`	Alternative program from label AUSW.
`N30 CS_IPMON("ALL");`	Enable break and overload
`N40 CS_TOOL("TOOL_7",1);`	Tool is replaced and IPM monitoring data activated.
`N50 S1000 M3 G1 F1000 X0 X100`	Machining is started.
	Monitoring event occurs!!! The program skips to the AUSW label through the alternative strategy.
`N70 GOTO END`	Normal program end if no monitoring event occurs.
`N80 AUSW:`	

N10	CS_ESCON	
N90	MSG ("Program skips to alternative label")	Programmer can intervene at this point.
N130	END:	
N140	....	
N150	MSG ("Program end")	
N160	M30	

1.2.6.5 Generating a log file

Scenario

Information for generating a log file in the alternative strategy. Under certain circumstances it may be advisable should a monitoring even occur (e.g. tool break) to document which tools fail during a shift.
For this purpose, a report can be produced using the "WRITE" instruction under the alternative label in the part program. More detailed information on file handling (WRITE, DELETE, etc.) can be found in the Siemens Programming Instructions for "Work Preparation".

Example

```
DEF INT _EC
DEF STRING[100] MTEXT
if(($P_Search<>1)AND($P_TOOLNO>0))
     MTEXT=" IPM BREAK MESSAGE, tool name: "
     <<$TC_TP2[$P_TOOLNO]<<" Duplo no.: "
     <<$TC_TP2[$P_TOOLNO]<<" Cutting edge: "<<$P_TOOL
     WRITE (_EC,
"LOGBOOK",$A_HOUR<<":"<<$A_MINUTE<< ":"<<$A_SECOND<<MTEXT)
     if(_EC==10)
        MSG("Logbook is full, deleted at NC Start!")
        m0
        DELETE (_EC,"LOG BOOK")
     endif
   endif ; if $P_SEARCH<>1
m30
```

1.2.6.6 Control override (from IPM1.2/9)

General

The special feature of control override is its ability to modify feedrate to the actual cutting conditions in realtime. If air cutting is detected, feedrate can in addition be multiplied by a fixed (can be revised through NC language command) factor. After an NC reset, this factor defaults to 150 (%).
This control feature enables optimal tool utilisation and prevents overload. As a result, production output can usually be increased.

Programming

In the IPM monitoring method:
Select "Control OVERRIDE without alarm" or "Control OVERRIDE with alarm" methods.
Three configuration parameters for the control algorithm:

- "upper override limit" (strategy parameter 1)
- "lower override limit" (strategy parameter 2)
- "Setpoint value and/or reference(output) to be controlled to (strategy parameter 3)

These parameters can be modified at any time via the NC language command `"CS_IPMMW("STRATEGY_PAR...)"`.
The "control override" method may occur in only one monitoring task. Otherwise, the one with the highest method group, method or task wins.

Further parameters can be modified if required through language commands:

- **Setpoint value (usually spindle rating) for air cutting detection:**
  Programming:
  `CS_IPMCO("FEEDRATE_AIR_CUT_LIMIT",x)`
  Initialisation after NC reset: x=0 (deactivated)
- **Override factor for air cutting detection:**
  Programming:
  `CS_IPMCO("FEEDRATE_AIR_CUT_FEED_FACTOR",x)` Initialised after NC reset: x=150
  (With default from MD N12030
  `$MN_OVR_FACTOR_FEEDRATE[30]` = 1.2 is max. 126% possible)

The values of the parameters below are optimised and should not be changed:

- **Time constants (Ti for I-controller and Tn for PI controller):**
  Programming:
  `CS_IPMCO("FEEDRATE_TIME",x)`
  Initialisation after NC reset: x=4s

- **Mean factor for the input signal:**
  Programming:
  `CS_IPMCO("FEEDRATE_AVERAGE_FACTOR",x)`
  Initialisation after NC reset: x=10
- **P-component (reinforcement for PI controller):**
  Programming:
  `CS_IPMCO("FEEDRATE_GAIN",x)`
  Initialisation after NC reset: x=0
  (deactivated = I controller is thus active)

Drain

The controller receives the actual value minus setpoint value differential (usually actual output minus reference output) as an input variable.
The variable is the override, which can only be influenced if the following preconditions are satisfied:

- "Control override" method selected
- Method active
- G0 not selected
- Override setting on machine control panel = 100%.

The output is controlled within the permissible limits (lower and upper override limits). This is usually handled by an I controller. When the lower override limit is reached, an alarm is generated provided the "Control OVERRIDE with alarm" option has been selected.

Note:
If you want the override to exceed a value of 126%, the machine datum N12030 $MN_OVR_FACTOR_FEEDRATE[30]=2 must be set. This is the maximum value. And allows a maximum of 210% of the programmed feed to be achieved.

Example

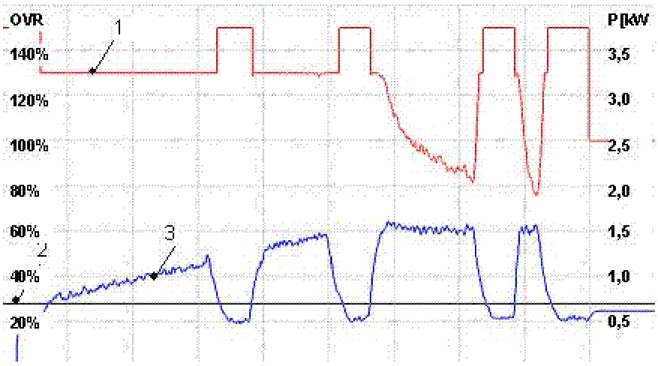
An example of an override control with air cutting detection is provided below. In this example, a workpiece is machined with a surface milling tool (oblique cut over three grooves).
The following parameters have been configured:

- upper override limit:	130 %
- lower override limit:	50 %
- Setpoint value (spindle rating):	1.5 KW
- Air cutting detection with:	0.7 KW
- Override factor for air cutting detection:	15 %

Graphic illustration

Standard for scaling:

- Override: 20 % / Div.
- Spindle rating: 0.5 KW / Div.



1 Override
2 Air cutting detection limit
3 Spindle power

The diagram shows that after target milling the second and third groove, the override is rolled back prevent the spindle rating exceeding 1.5 kW and keep it at no more than this permissible rating. Furthermore, if the air cutting detection limit is exceeded, the override is raised to 130% (top override limit) in order to optimise tool utilisation.
When the milling cutter is cutting air, the override is raised to 150% to minimise non-cutting time.

1.2.6.7 Alarm signals

75120 "Channel %1 IPM interpreter configuration alarm error no.: %2

Error number:

1: "CC_IPM_CONFIG_ERROR_MEM"

The IMP compiler cycle has insufficient memory.
Remedy: Increase MD62803, possibly also MD28105 (Cond.: MD62803<=MD28105)

2: "CC_IPM_CONFIG_ERROR_MD"

The IPM compiler cycle has problems creating machine data.

Under normal circumstances this should never occur on the machine.

3: "CC_IPM_CONFIG_ERROR_CC_BLOCK_ELEMENTS_USER_MEM"

The memory for the block element is too low.
Remedy: Increase MD28090 and/or 28100.

4: "CC_IPM_CONFIG_ERROR_IPO_SYSCLOCK_TIME_RATIO"

The IPO servo cycle ratio may not exceed 3.
Remedy: MD10070 must be either 1, 2 or 3.

5: "CC_IPM_CONFIG_ERROR_MM_NUM_CC_MON_PARAM"

Parameters $TC_MOPC1 and $TC_MOPC2 are not present.
Remedy: MD18099 $MN_MM_NUM_CC_MON_PARAM>=2
MD18080 $MN_MM_TOOL_MANAGEMENT_MASK set Bits 0 and 2.

6: "CC_IPM_CONFIG_ERROR_MM_NUM_CC_TDA_PARAM"

Parameters $TC_DPC9 and $TC_DPC10 are not present.
Remedy: MD18094 $MN_MM_NUM_CC_TDA_PARAM>=10
MD18080 $MN_MM_TOOL_MANAGEMENT_MASK set Bit 2.

7: "CC_IPM_CONFIG_ERROR_NCK_RESET_REQUIRED"

Another NCK reset is required.
Remedy:
Press NCK reset through commissioning main menu, reset keys function on the NCU or switch machine off/on.

75121 "Channel %1 block %2 IPM Interpreter programming error no.: %3

Error number:

1: "CC_IPM_ERROR"

This is a general error which should actually never occur.

2: "CC_IPM_ERROR_PARA_STRING_NOT_FOUND"

The transfer parameter specified in the instruction is not present.
Remedy: Check spelling and/or syntax.

3: "CC_IPM_ERROR_PARA_STRING_NOT_ALLOWED"

The transfer parameter specified in the instruction is not permitted in this context.
Remedy: Check spelling and/or syntax.

4: "CC_IPM_ERROR_PARA_METHOD_NR"

The specified method number lies outside of the valid value range.
Values 1 and 2 are currently allowed.
Remedy: enter value method number.

5: "CC_IPM_ERROR_PARA_TASK_NR"

The specified task number lies outside of the valid value range.
Values 1 and 2 are currently allowed.
Remedy: Enter valid task number

6: "CC_IPM_ERROR_READ_FIRST_STACK_ARGUMENT"

Error occurs when the first transfer parameter is being read.
This is a general error which should actually never occur.

7: "CC_IPM_ERROR_READ_SECOND_STACK_ARGUMENT"

Error occurs when the second transfer parameter is being read.
This is a general error which should actually never occur.

8: "CC_IPM_ERROR_REA_BLOCKELEMENT_FAILED"

Error on reading block element.
This is a general error which should actually never occur.

9: "CC_IPM_ERROR_CC_BLOCK_USER_MEM"

The memory for the block element is too low.
Remedy: Increase MD28090 and/or 28100.

10: "CC_IPM_ERROR_CC_PUSH_RETURNVALUE"

Error as a value is being returned to the CC function.
This is a general error which should actually never occur.

11: "CC_IPM_ERROR_GUD_METHODS_NOT_FOUND"

The methods data of the tool cannot be found in the SGUDs.
Remedy: Correct method reference or execute management comparison of tool data.

12: "CC_IPM_ERROR_EVENT_NOT_ALLOWED"

The projected monitoring event is not possible.
This is a general error which should actually never occur.

13: "CC_IPM_ERROR_JOB_MASK_NOT_ALLOWED"

The selected monitoring tasks are not present.
This is a general error which should actually never occur.

14: "CC_IPM_ERROR_JOB_CONNECT NOT_ALLOWED"

The specified connection is not possible.
This is a general error which should actually never occur.

15: "CC_IPM_ERROR_ALARM_RESPONSE_NOT_ALLOWED"

The specified alarm response is not possible.
This is a general error which should actually never occur.

16: "CC_IPM_ERROR_STRATEGY_NOT_ALLOWED"

The specified monitoring strategy is not possible.
This is a general error which should actually never occur.

17: "CC_IPM_ERROR_GUD_READ_BINDING"

The GUDs cannot be read.
This is a general error which should actually never occur.

18: "CC_IPM_ERROR_GUD_METHOD_SYMBOL_NOT_FOUND"

The GUDS to be read are not present.
This is a general error which should actually never occur.

19: "CC_IPM_ERROR_READ_THIRD_STACK_ARGUMENT"

Error occurs when the third transfer parameter is being read.
This is a general error which should actually never occur.

20: "CC_IPM_ERROR_METHOD_GROUP_NOT_ALLOWED"

The specified method group is not present or not configured.
Remedy: Correct NC program or increase MD6284.

21: "CC_IPM_ERROR_SENSOR_ID_TYPE_NOT_ALLOWED"

The specified sensor ID (sensor type) is not possible.
This is a general error which should actually never occur.

22: "CC_IPM_ERROR_READ_FOURTH_STACK_ARGUMENT"

Error occurs when the fourth transfer parameter is being read.
This is a general error which should actually never occur.

23: "CC_IPM_ERROR_SENSOR_ID_MACH_AX_INDEX_NOT_ALLOWED"

The specified sensor ID (sensor location) is not possible.
This is a general error which should actually never occur.

24: "CC_IPM_ERROR_SENSOR_ID_CHAN_AX_NOT_ALLOWED"

The specified sensor ID (sensor location) is not possible.
This is a general error which should actually never occur.

25: "CC_IPM_ERROR_SENSOR_ID_CHAN_AX_INDEX_FROM_AXTAB_NOT_ALLOWED"

The specified sensor ID (sensor location) is not possible.
This is a general error which should actually never occur.

26: "CC_IPM_ERROR_SENSOR_ID_GEO_CHAN_AX_NOT_ALLOWED"

The specified sensor ID (sensor location) is not possible.
This is a general error which should actually never occur.

27: "CC_IPM_ERROR_SENSOR_ID_GEO_AX_INDEX_FRROM_AXTAB_NOT_ALLOWED"

The specified sensor ID (sensor location) is not possible.
This is a general error which should actually never occur.

75122 "Channel %1 IPM fracture alarm, method no.: %2, method ID: %3%4

75126 "Channel %1 IPM fracture alarm (cut), method no.: %2, method ID: %3%4

75128 "Channel %1 IPM fracture alarm (missing), method no.: %2, method ID: %3%4

Method no.: %2

1. If the method was input via the control panel (MMC), the value 1 appears in method group 1 with an overload alarm. If the event was defined through the programming instruction, method 1 of method group 1 should be set to the "break" monitoring event. Program example: `CS_IPMMW ("EVENT[1]",1,1)`
2. If the event was defined through the NC programming instruction, method 2 of method group 1 has been set to the "break" monitoring event. Program example: `CS_IPMMW ("EVENT[2]",1,1)`
3. If the method was input via the control panel (MMC), the value 2 appears in method group 3 with an overload alarm. If the event was defined through the programming instruction, method 2 of method group 2 should be set to the "break" monitoring event. Program example: `CS_IPMMW ("EVENT[1]",1,2)`
4. If the event was defined through the NC programming instruction, method 2 of method group 2 has been set to the "break" monitoring event. Program example: `CS_IPMMW ("EVENT[2]",1,2)`

Methods ID: %3

This is where the method ID is specified, which is automatically assigned during generation of a monitoring method.
This is used to identify the method.
If the value "0" is specified, the method can no longer be identified, since it has been changed by the NC language instruction. It no longer corresponds to the method in the database (MMC).

%4

If software status >= V1.1/5 is used, an additional string that is transferred on method selection (usually cutting tip or tool change) will be output.
Usually, the tool identifier, the Duplo number and the active cutting tip number are displayed.
This identifies the tool that has triggered the alarm.

75123 "Channel %1 IPM overload alarm, method no.: %2, method ID: %3%4

75127 "Channel %1 IPM overload alarm (blunt), method no.: %2, method ID: %3%4

Method no.: %2	1. If the event was defined through the NC programming instruction, method 1 of method group 1 has been set to the "overload" monitoring event. Program example: `CS_IPMMW ("EVENT[1]",2,1)` 2. If the method was input via the MMC interface, the value 2 always appears with an overload alarm. If the event was defined through the NC programming instruction, method 2 of method group 1 has been set to the "overload" monitoring event. Program example: `CS_IPMMW ("EVENT[2]",2,1)` 3. If the event was defined through the NC programming instruction, method 1 of method group 2 has been set to the "overload" monitoring event. Program example: `CS_IPMMW ("EVENT[1]",2,2)` 4. If the method was input via the MMC interface, the value 4 always appears with an overload alarm. If the event was defined through the NC programming instruction, method 2 of method group 2 has been set to the "overload" monitoring event. Program example: `CS_IPMMW ("EVENT[2]",2,2)`
Methods ID: %3	This is where the method ID is specified, which is automatically assigned during generation of a monitoring method. This is used to identify the method. If the value "0" is specified, the method can no longer be identified, since it has been changed by the NC language instruction. It no longer corresponds to the method in the database (MMC).
%4	If software status >= V1.1/5 is used, an additional string that is transferred on method selection (usually cutting tip or tool change) will be output. Usually, the tool identifier, the Duplo number and the active cutting tip number are displayed. This identifies the tool that has triggered the alarm.

75124 "Channel %1 IPM Interpolator alarm error no.: %2

Error number:

1: "CC_IPM_IPO_CONFIG_ERROR"

This is a general error in the interpolator which should actually never occur.

2: "CC_IPM_IPO_CONFIG_ERROR_ACMARKERAREA"

There is no access to the required marker area. The marker is presumably not fitted.
Remedy: check MD N28256 $MC_MM_NUM_AC_MARKER.

75125 "Channel %1 IPM servo alarm error no.: %2

Error number:

No error numbers have yet been assigned, in other words, the error cannot yet occur.

75129 "Channel %1 IPM General error no.: %2%3%4"

Error number:

1: "CC_IPM_GENERAL_ERROR_NO_MORE_MEMORY"

The free NCK memory is quickly used up when files are written.
Remedy: Remove the unrequired parts of the program.

2: "CC_IPM_GENERAL_ERROR_ASUP_NOT_AVAILABLE"

The ASUP specified through $MC_IPM_ASUP_FILE_NAME could not be logged on.
Remedy: Correct program name or load program.

75130 "Axis %1 IPM collision alarm%2%3%4"

Error number:

Variables %2, %3 and %4 have not yet been used.

75131 "Axis %1 IPM belt break alarm%2%3%4"

Error number:

Variables %2, %3 and %4 have not yet been used.

75149 "%1%2%3%4"

This alarm is used for alarm outputs from the NCK-OEM development. The complete text is present in the NCK-OEM software. If an alarm occurs, it will relate to a system error.