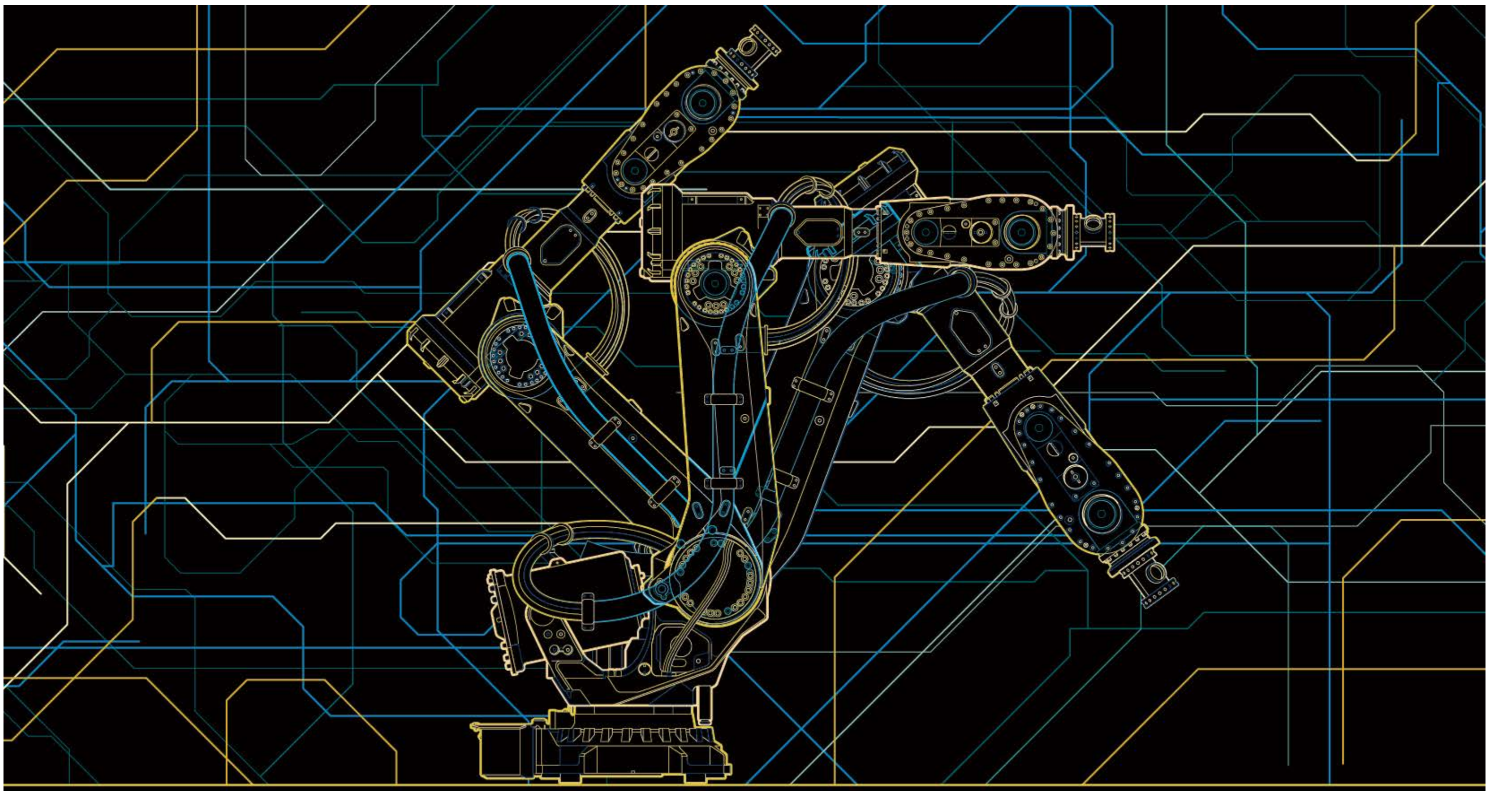


Application manual
Production Monitor

Power and productivity
for a better world™

Trace back information:
Workspace RW 6-0 version a3
Checked in 2014-11-11
Skribenta version 4.1.349

Application manual
Production Monitor

RobotWare 6.0

Document ID: 3HAC050963-001
Revision: -

ABB AB

Robotics Products

Se-721 68 Västerås

Sweden

Table of contents

1 Introduction to Production Monitor

Introduction

Production Monitor is an Arc sub-option that enables statistical and event data logging to WebWare databases and/or comma-delimited text files. Production Monitor is also capable of providing immediate operator notification of production problems.

Continues on next page

1.1 Product overview

1.1.1 About Production Monitor

Product overview

Data concerning production events and part completion status are written to custom tables in a WebWare 4.6, or higher, database. Arc statistics such as unscheduled stops, skipped welds, and accumulated arc time are stored in database tables specific to seam-level details and broad cycle-level overviews.

Production Monitor may be configured to provide immediate notification dialogs to the operator in the event that a part contains incomplete weld seams. The notification may be in the form of FlexPendant dialogs, digital outputs, or both.

In addition to Arc-related data, Arc sub-options also maintain tables of their own. BullsEye, SmarTac, Torch Service Center, and Navigator, all maintain tables with time-stamped results

Application guide

Production Monitoring provides basic statistical data logging for Arc, Production Manager, BullsEye, Navigator, Torch Service Center, and SmarTac.

Cycle data is available only when Production Manager is installed and its Execution Engine used. Cycle data may also be provided by alternative execution engines using the RAPID interface provided by Production Monitor. More information regarding the RAPID interface can be found in the chapter *RAPID instructions on page 31*.

Production Monitor does not monitor arc characteristics. It only monitors starts and stops during weld seams and the resulting seam length. It evaluates a seam's success based on a nominal seam length stored for the seam. It evaluates a part's success based on the success of all the seams included in the part.

1.1.2 Operation overview

Operation overview

Production Monitor is easy to use. Once it is set up, the database-writing features are transparent to the operator.

WebWare database design

If the WebWare writing is configured, the data will be stored in a database called *ProdMonDB*. The following tables will be available if all arc sub-options are installed:

WebWare database: ProdMonDB

RobotWare option	Table name	Description
Arc	tblSeamEv	Seam events
Arc	tblSeamRes	Seam results
Arc	tblCycleRes	Cycle results
Production Manager	tblGapEv	GAP Execution Engine events
Production Manager	tblCycleEv	GAP Execution Engine part cycle events
BullsEye	tblBullsEye	BullsEye results
SmarTac	tblSmtc1D	Search1D results
SmarTac	tblSmtcPart	SearchPart results
SmarTac	tblSmtcGrv	SearchGroove results
Torch Service Center	tblTchClean	Torch Cleaner results
Navigator	tblNavSrch	Search Sphere results
Navigator	tblNavMeas	Measure 1D results

Comma-delimited file names

If Production Monitor is configured to write to comma-delimited text files on the robot controller's drive, the files are stored, by default, in a temp directory under the system directory. The file names will be as follows:

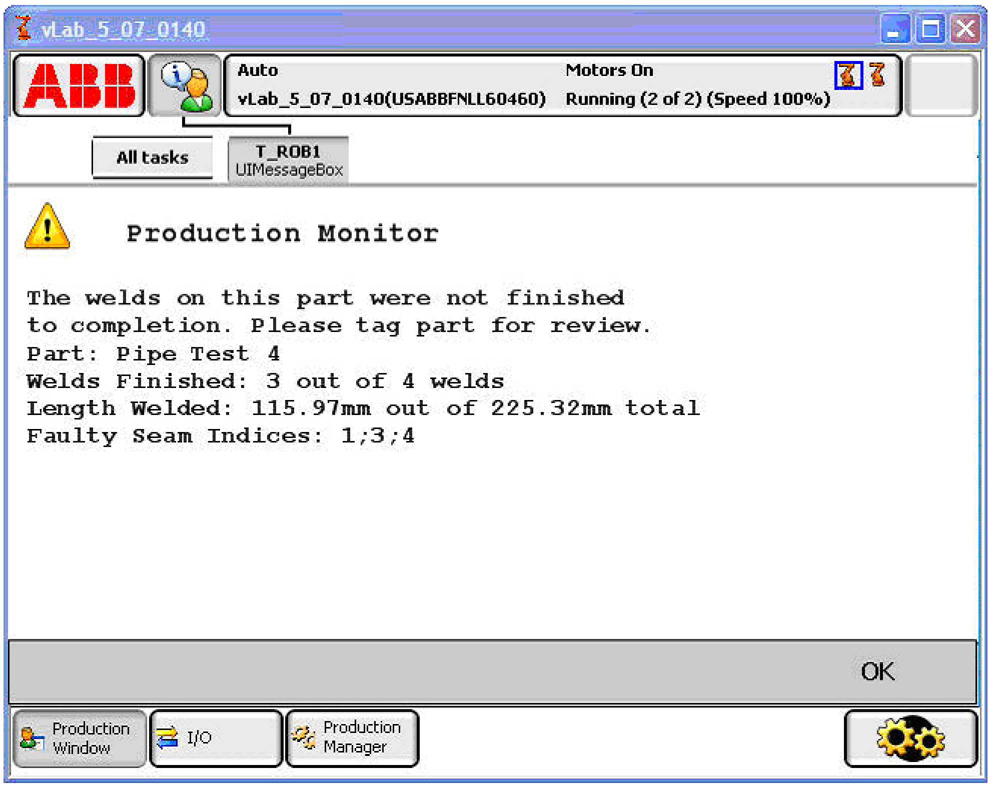
RobotWare option	File name	Description
Arc	Not available*	Seam events
Arc	SeamResults.csv	Seam results
Arc	CycleResults.csv	Cycle results
Production Manager	GapEvent.csv	GAP Execution Engine events
Production Manager	CycleEvent.csv	GAP Execution Engine part cycle events
BullsEye	BullsEye.csv	BullsEye results
SmarTac	Search1D.csv	Search1D results
SmarTac	SearchPart.csv	SearchPart results
SmarTac	SearchGroove.csv	SearchGroove results
Torch Service Center	TchService.csv	Torch Cleaner results

Continues on next page

RobotWare option	File name	Description
Navigator	SearchSp.csv	Search Sphere results
Navigator	Measure1D.csv	Measure 1D results

Seam fault notification

Production Monitor may be configured to notify the operator in the event a weld seam is incomplete. The notification may be in the form of a pop-up dialog box on the FlexPendant, or via digital outputs.



Production Monitor

```
The welds on this part were not finished
to completion. Please tag part for review.
Part: Pipe Test 4
Welds Finished: 3 out of 4 welds
Length Welded: 115.97mm out of 225.32mm total
Faulty Seam Indices: 1;3;4
```

Pop-up-Fault

Digital outputs may be configured to turn on indicator lamps or notify a PLC of the faulty condition. The I/O interface provides for two-way handshaking allowing a PLC to acknowledge the problem and allow execution to continue.

The I/O interface may be used in conjunction with the FlexPendant dialogs. Either notification method may be enabled independently from the other.

1.2 Requirements overview

System prerequisites

- Controller IRC5.
- RobotWare: 5.07.01 or higher with Arc and Production Monitor Arc sub-option.
- Optional Arc sub-options: BullsEye, Torch Service Center, SmarTac, and Navigator.
- WebWare Server 4.6 or higher with Reports Module, if configured to write to WebWare database.

Users qualifications

- **Operator:** No additional training required.
- **WebWare Server Administrator:** Administrator must be comfortable managing WebWare Server, including administration tasks associated with WebWare's Custom Database features.
- **Data Analyst:** Users viewing Production Monitor data must be familiar with WebWare Client and be able to generate reports using the Reports feature in WebWare.

This page is intentionally left blank

2 Installation

2.1 Components

Components list

Production Monitor is a RobotWare Arc sub-option. Purchasing the option provides the necessary components in the robot controller to perform the monitoring functions. The option does not include WebWare Server or the hardware required to run WebWare Server.

Hardware

Please refer to WebWare documentation for help setting up a WebWare server.

2.2 Software installation

Selecting in Installation Manager

Production Monitor is sub-option to *RobotWare Arc*. If purchased, the controller key will make the option available. Be sure to select the option when building a system with **Installation Manager**.

Creating the database in the WebWare server using RAPID

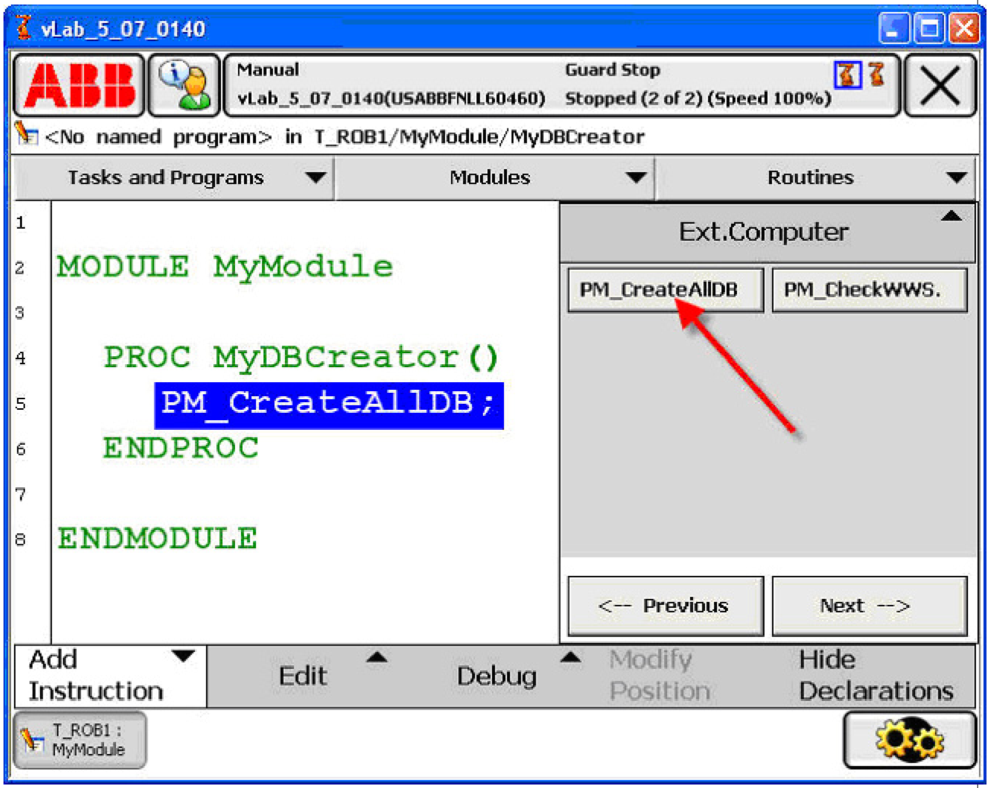
This section is only applicable to systems using the WebWare Server interface.

To create the WebWare database for Production Monitor, a RAPID instruction has been provided to create the database and tables from a robot controller.

After ensuring that WebWare is able to connect to the robot controller, create a simple RAPID routine that makes a call to the instruction, `PM_CreateAllDB`.

Note

If you received a turn-key software package from ABB for your particular system, you may be able to access a menu item in the Production Manager Setup Menu that is pre-defined to call the `PM_CreateAllDB` instruction. If so, refer to the documentation included with the system.



```
MODULE MyModule

  PROC MyDBCreator()
     PM_CreateAllDB;
  ENDPROC

ENDMODULE
```

Call-PM_Crea

Move the program pointer to the newly created routine and run the routine in Automatic mode for one cycle. This instruction will walk through each of the Arc sub-options and Production Manager option attempting to create the appropriate

Continues on next page

database tables listed in section *WebWare database design on page 9*. Messages will be written to the Process Error Logs if the table creation was successful, or if it failed. In either case, the name of the table will be included in the message. If the table already existed, nothing will be written to the log.

Note

Many robot controllers may write to a single WebWare Server. The database tables need only be created from one robot controller to make it available to all the others.

Creating the database in the WebWare server using a script

Alternatively, the ProdMonDB database and tables may be created using a script tool on the WebWare Server itself. Although, not available at the time of printing, a server tool may be available in the future that supports this feature. Please refer to the documentation included with the script files for more information regarding this new feature.

2.3 Start-up test

Start-up test

Once the software has been installed and the database created in WebWare Server, if applicable, the robot controller should be able to write to the tables. To test, create a routine with a weld seam. Be sure to use the `SeamName` optional argument in the `ArcStart` instruction to ensure that you can identify the seam. With the system in Automatic mode, run the routine. After the routine is complete, there should be new records in the Seam Events table (tblSeamEv). If not, refer to the WebWare documentation to determine why the controller is unable to write to the database.

Only the Seam Events table (tblSeamEv) is written to after each seam event. The other tables provided by Arc and Production Manager are written to only after a part cycle is complete. More information can be found about this in the sections that follow.

3 Using Production Monitor

3.1 Introduction

Introduction

This chapter provides practical instruction to configure Production Monitor.

3.2 Evaluation of Part Completion

Overview

Arc seam length is evaluated against saved nominal seam length plus a comparison factor. Part completion is determined by the length of each seam in the part and the overall number of welds. Production Monitor reports the results of the evaluation to the WebWare tables, comma-delimited text files, pop-up dialog boxes, and digital outputs, depending on what Production Monitor features are enabled.

Length comparison factor

Production Monitor designates a seam as complete if the recorded seam length is at least as long as the saved nominal seam length with a margin of error specified by the *comparison factor*. The comparison factor is specified in the Process configuration database, sometimes referred to as the *PROC*.

```
PROC:CFG_1.0:5:0::
# created 2005/06/07
#
ARC_PROD_MONITOR:
-name "default" -len_compare_factor 0.98
```

The default is set to 0.98, which implies that the measured seam length must be at least 98% of the nominal length to be considered "complete".

Number of welds

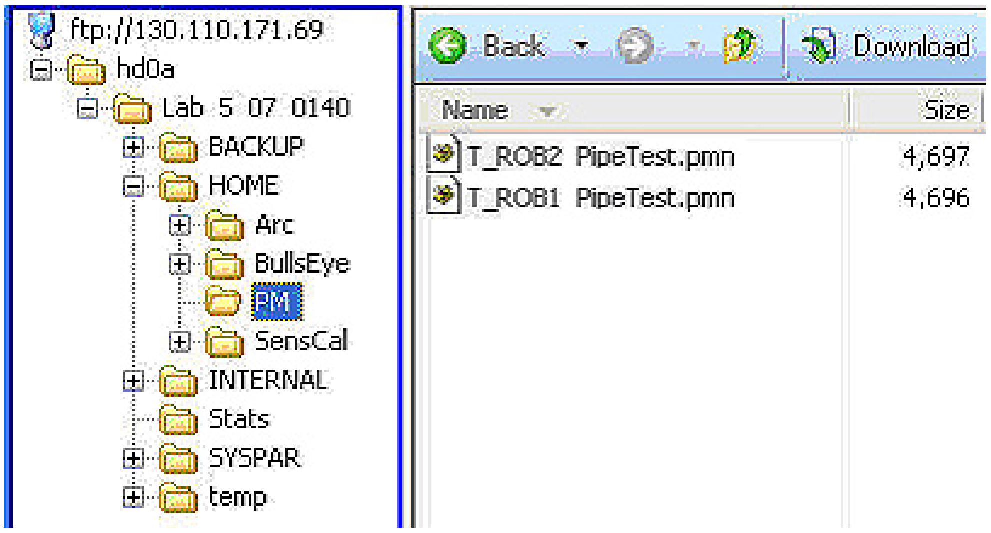
Production Monitor stores the number of welds present in each part. If a part cycle ends and the number of seams recorded is less than the number of nominal seams, Production Monitor will consider the part incomplete. If the number of welds recorded is greater than the nominal amount, an error message will inform the user that the nominal data should be discarded and re-initialized.

Continues on next page

Nominal data storage

Nominal data for each seam is stored in a separate file for each part. Production Monitor will only make seam length comparisons for parts routines that have corresponding partdata instances executed by the Production Monitor Execution Engine.

When a part is run for the first time, a file is stored on the robot's hard drive that contains nominal information for each weld seam in the part. The file is stored under the robot controller's HOME directory:



Name	Size
T_ROB2 PipeTest.pmn	4,697
T_ROB1 PipeTest.pmn	4,696

Nominal-data

When subsequent parts are run, Production Monitor will use the saved data for the evaluation. It is recommended that the initial data be stored while executing in Automatic mode to ensure that the stored data is representative of actual production speeds. If a change is made to the part routine, the data file should be deleted from the PM directory so that a new file is stored on the next part cycle.

Note

Be sure to use the optional SeamName argument in the `ArcStart` instruction for all weldseams to be monitored by Production Monitor. Otherwise the seam can not be identified by Production Monitor, and no nominal files are saved.

Moving the Program pointer (skipping weld seams)

If the PP is moved from a weld seam, the Cycle Results table will show that seam as having zero length. If part of the seam was completed, Production Monitor will not log that information. There will, however, be an entry in Seam Events table that shows the stop position prior to moving the PP.

3.3 Data logging

Overview

Production Monitor is capable of sending data to a WebWare database and/or comma-delimited text files on the robot controller's hard drive.

WebWare server

Ideally, WebWare should be used as a data store, rather than the comma-delimited text files. The WebWare database provides a reporting tool that allows the user to filter the queried results and build reports that include only the data relevant to the task at hand. When WebWare Server is configured with Microsoft SQL Server as a data store, large amounts of data can be stored for future analysis.

Enabling

WebWare writing is configured in the Process configuration database (PROC). Access can be reached through RobotStudio, or via config files written with proper syntax.

```
PROC:CFG_1.0:5:0::
# created 2005/06/07

#
PM_SETTINGS:

-name "Standard" -log_to_ww -log_to_csv
```

Manual teach mode considerations

If Production Monitor is configured to write to a WebWare database, it will only do so in automatic mode, by default. It will not attempt to write in manual mode, or manual full speed mode. This is due to a limitation in WebWare 4.X versions.

Future WebWare versions may allow writing in manual mode. Should the version you are using support this, Production Monitor may be configured to write in manual mode by enabling the option in the PROC configuration.

```
PROC:CFG_1.0:5:0::
# created 2005/06/07

#
PM_SETTINGS:

-name "Standard" -log_to_ww -ww_in_manual
```

Communication errors

If the controller cannot make contact with the WebWare server, no errors will be reported by the robot controller. When the connection is re-established, writing will resume. Consult the "timers" in the WebWare Client to determine when a controller was offline. More information about this can be found in the WebWare documentation.

Continues on next page

Comma-delimited text files

The comma-delimited files are typically viewed through a spreadsheet program like Microsoft Excel that does not support queries. The files are provided mainly as diagnostic tools for debugging problems in the WebWare Server link to the controller. The comma-delimited files are limited in size. As a result, they are automatically purged when they grow too large. No warning is given that this is about to happen. The data is not archived in any way.

Enabling

Comma-delimited files are enabled in the Process configuration database (PROC). Access can be reached through RobotStudio, or via config files written with proper syntax.

```
PROC:CFG_1.0:5:0::
# created 2005/06/07

#
PM_SETTINGS:

-name "Standard" -log_to_ww -log_to_csv
```

File path

The path to the comma-delimited files is configurable. By default, the files are stored under the robot controller's HOME directory in a directory called "temp". However, the path may be overridden to include any location on the hard drive.

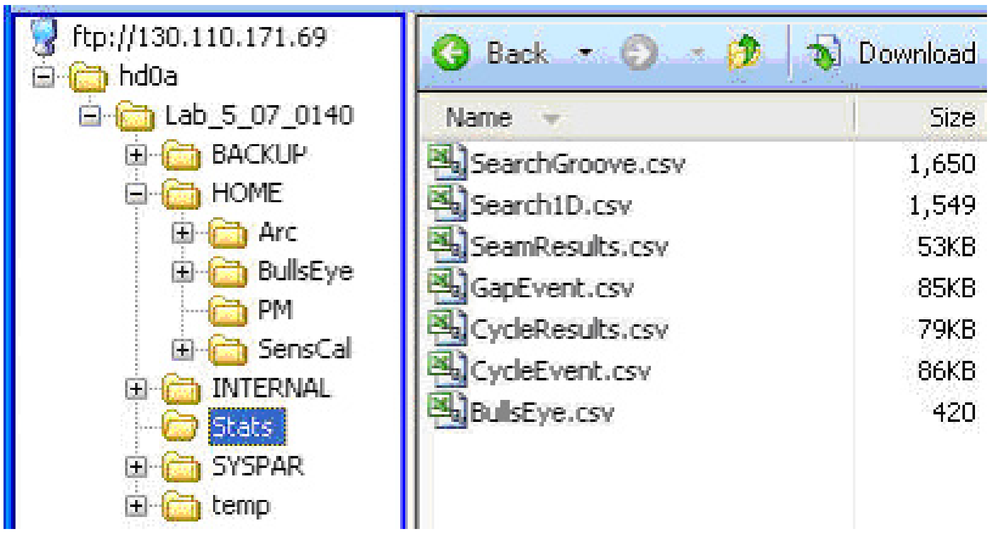
```
PROC:CFG_1.0:5:0::

# created 2005/06/07

#
PM_SETTINGS:
-name "Standard" -log_to_csv \
-csv_log_path "../Stats/"
```

Continues on next page

The example above shows a path to the directory displayed below:



ftp://130.110.171.69
- hd0a
  - Lab_5_07_0140
    - BACKUP
    - HOME
      - Arc
      - BullsEye
      - PM
      - SensCal
    - INTERNAL
    - Stats
    - SYSPAR
    - temp

Name	Size
SearchGroove.csv	1,650
Search1D.csv	1,549
SeamResults.csv	53KB
GapEvent.csv	85KB
CycleResults.csv	79KB
CycleEvent.csv	86KB
BullsEye.csv	420

Directory-Co

Performance

A large amount of data is collected while welding. To ensure that cycle time is not affected, the data is written to WebWare and the comma-delimited files after a part cycle is complete. This allows to the controller to pass the information while the robot is moving between part cycles. The exception to this rule is the Seam Events WebWare table (tblSeamEv). Data records are written to this table whenever the welding process starts or ends. Due to performance limitations of the file system, only WebWare supports the Seam Events table. This table is not included in the comma-delimited files.

Note

Do not attempt to open the comma-delimited files with editors like Excel when running production on the Virtual Controller. Doing so causes file access errors. Copy the file to another location before opening.

EventID and CycleID

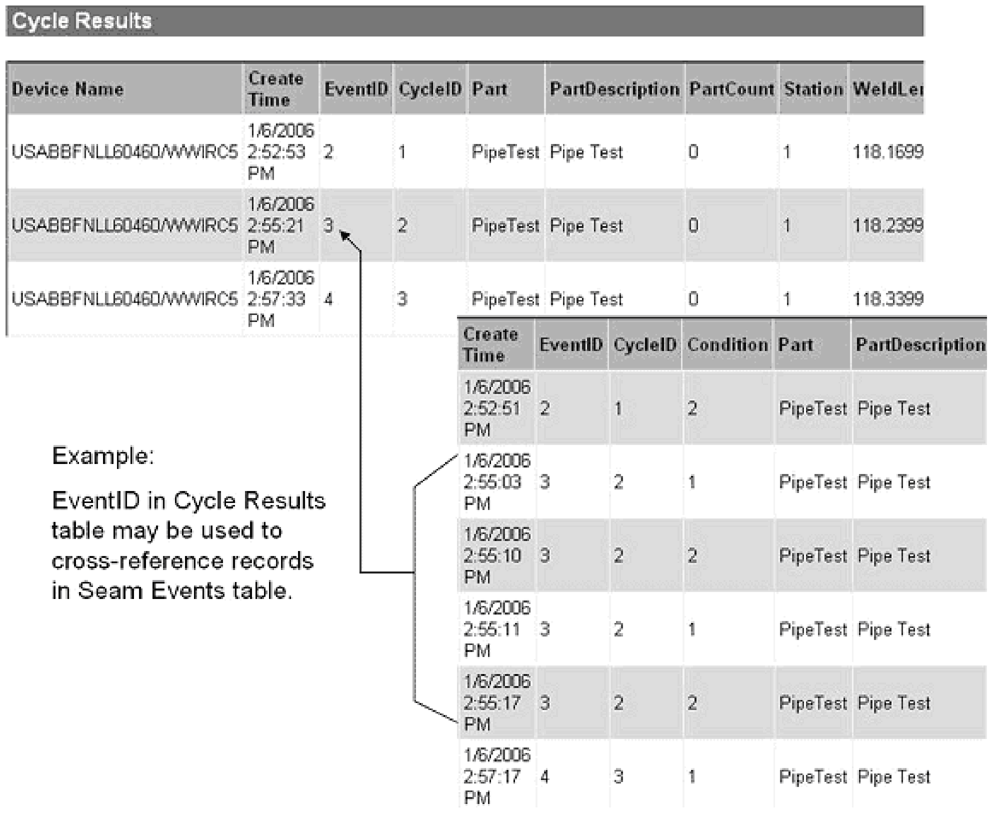
Every data table created by Production Monitor contains two fields that may be used to cross reference data between multiple tables. These fields are called keys. The keys are intended to be semi-unique for a given robot controller and motion task. That is to say, there could be duplicate values found in the EventID and CycleID columns, but not many. By default, the Production Manager Execution Engine will provide unique, incrementing values for the fields unless the following occurs:

- At restart using the restart modes **Reset RAPID**, **Reset system**, or **Revert to last auto saved**
- The numbers grow very large (99,999)

In either case, the values will resume at 1.

Continues on next page

The EventID field provides an incrementing key for all Execution Engine events including part execution and service routine execution. The CycleID provides an incrementing key for part cycles, only.



Cycle Results

Device Name	Create Time	EventID	CycleID	Part	PartDescription	PartCount	Station	WeldLe
USABBFNLL60460/WWIRC5	1/6/2006 2:52:53 PM	2	1	PipeTest	Pipe Test	0	1	118.1699
USABBFNLL60460/WWIRC5	1/6/2006 2:55:21 PM	3	2	PipeTest	Pipe Test	0	1	118.2399
USABBFNLL60460/WWIRC5	1/6/2006 2:57:33 PM	4	3	PipeTest	Pipe Test	0	1	118.3399

Create Time	EventID	CycleID	Condition	Part	PartDescription
1/6/2006 2:52:51 PM	2	1	2	PipeTest	Pipe Test
1/6/2006 2:55:03 PM	3	2	1	PipeTest	Pipe Test
1/6/2006 2:55:10 PM	3	2	2	PipeTest	Pipe Test
1/6/2006 2:55:11 PM	3	2	1	PipeTest	Pipe Test
1/6/2006 2:55:17 PM	3	2	2	PipeTest	Pipe Test
1/6/2006 2:57:17 PM	4	3	1	PipeTest	Pipe Test

Example:

EventID in Cycle Results table may be used to cross-reference records in Seam Events table.

Example-Even

Both the EventID and CycleID are accessible from RAPID at anytime by querying the instructions, `GapGetCycleID` and `GapGetEventID`. More information about these instructions can be found in chapter *RAPID instructions on page 31*.

Accumulated arc time and arc starts

Production Monitor for Arc will track accumulated arc-on time and the number of arc starts. This information could be useful for tracking welding consumable degradation and maintenance requirements. The data is reported to the Cycle Results table and file (tblCycleRes, CycleResults.csv).

Accumulated arc time is recorded in seconds. The values for both the time and number of starts will resume at zero, if the values are allowed to grow too large. The values can be reset using the RAPID instruction, `ResetArcAccum`.

The values may be retrieved at any time using the RAPID instruction, `GetArcAccum`. The instruction retrieves both the accumulated arc time and the number of arc starts. This information could be useful to automatically run the welding torch cleaner after a certain number of arc starts, for example.

For more information regarding `ResetArcAccum` and `GetArcAccum`, please refer to chapter *RAPID instructions on page 31*.

Continues on next page

Part counter

PartCount is a field in CycleResults that may be used to track the number of parts that the system has produced. The logic for this counter must be supplied by the designer of the system.

Production Manager execution engine

Production Manager includes a part counting mechanism. The information supplied to this mechanism will be included in the CycleResults data. Two instructions are provided to grant read and write access to the part counter for each part. These are `GetCurrPartCount` and `SetCurrPartCount`. The part count value will be stored in the Production Manager Execution Engine for the duration of the part cycle only. It is up to the designer of the system logic to maintain the part count values for each part in RobotWare version 5.07.

Below is a simple example showing a basic part counter that is shared by all parts in the task.

```
%%%
VERSION:1
LANGUAGE:ENGLISH
%%%

MODULE SystemLogic(SYSMODULE)
CONST menudata mdClearPartCount :=
["Clear Part Counter",
"MyImage.gif",
"ClearPartCount", 3,
"", GAP_SHOW_ALWAYS,
TRUE, GAP_SERVICE_TYPE,
255, FALSE,0];

LOCAL VAR num nPartCount;

PROC BeforePart()
SetCurrPartCount nPartCount;
ENDPROC

PROC AfterPart()
GetCurrPartCount nPartCount;
ENDPROC

PROC ClearPartCount()
nPartCount:=0;
ENDPROC

ENDMODULE
```

The Production Manager Execution Engine will increment the part counter if the part is finished. `GetCurrPartCount` allows the system logic to update its local part counter variable.

Continues on next page

Custom execution engine

Systems that do not use the Production Manger Execution Engine must provide specific information to Production Monitor to allow it to report data related to cycles. The part count is no exception. Production Monitor provides two instructions that are used to inform the option that a part cycle is starting and ending. These are `ProcCycleStart` and `ProcCycleEnd`.

More information about these instructions can be found in chapter *RAPID instructions on page 31*, as well as in section *Production Monitor without Production Manager on page 28*.

3.4 Part incomplete notification

Overview

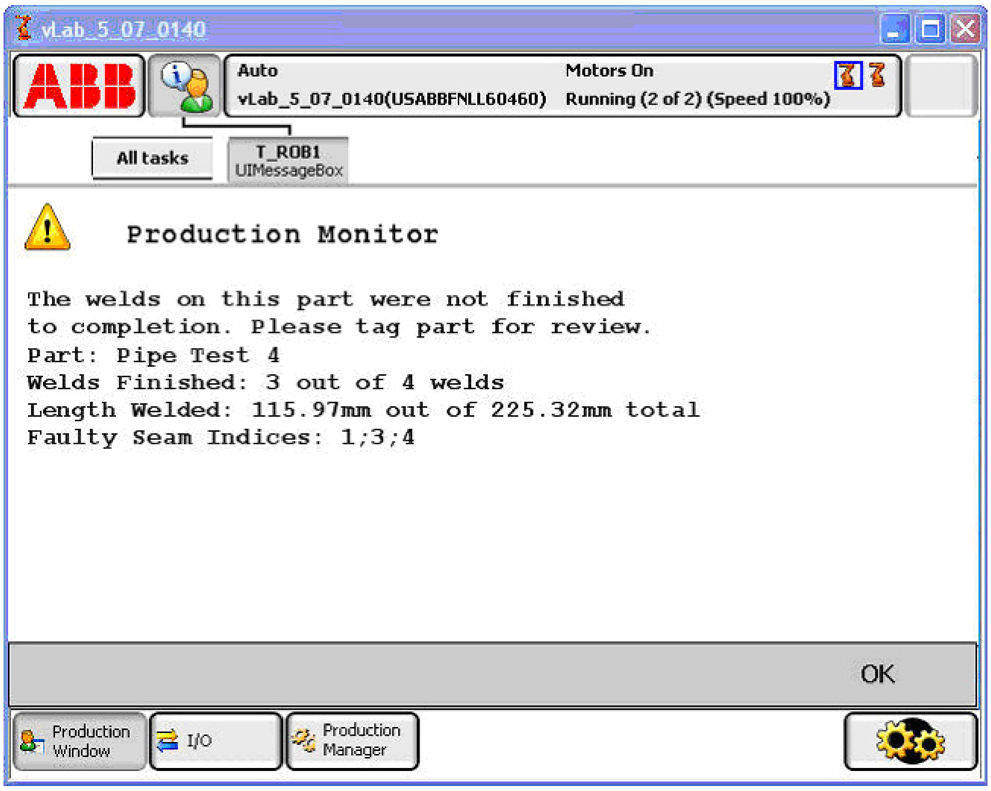
Production Monitor provides two notification tools to report incomplete parts.

- FlexPendant pop-up dialog box
- Configurable digital signals

Each may be enabled independently of the other.

FlexPendant pop-up dialog box

If configured, Production Monitor will launch a dialog box at the end of the part cycle and pause production until the user acknowledges that the part was not finished to completion.



```
Production Monitor

The welds on this part were not finished
to completion. Please tag part for review.
Part: Pipe Test 4
Welds Finished: 3 out of 4 welds
Length Welded: 115.97mm out of 225.32mm total
Faulty Seam Indices: 1;3;4
```

Dialog-Incom

Configuration

This functionality may be enabled in the Process configuration database (PROC).

```
PROC:CFG_1.0:5:0::
# created 2005/06/07

#
PM_SETTINGS:
-name "Standard" -log_to_ww \
-notify_at_cyc_end
```

Continues on next page

Configurable digital signals

Digital I/O may be configured to drive indicator lamps or to communicate with another device, such as a PLC.

Outputs

There are two digital outputs for each task that could be used to turn on an indicator lamp or notify an overseeing PLC.

- Part complete
- Part incomplete

The state of the signals is updated at cycle end. The signals will be automatically reset:

- at the start of the next cycle
- when the Reset Indicators input goes high
- when the optional pendant pop-up dialog is acknowledged.

Inputs

There is one digital input for each task that serves as the reset for the status outputs.

- Reset indicators

This signal should be used when the outputs are being monitored by an overseeing PLC. The signal should be set high by the PLC after the operator has acknowledged an incomplete status.

Configuration

The following PROC file listing demonstrates the IO interface for Production Monitoring:

```
PROC:CFG_1.0:5:0::
# created 2005/06/07

#
PM_SETTINGS:
-name "Standard" -log_to_ww \
-complete_indicator_1 "doPartComp1" \
-incomplete_indicator_1 "doPartIncomp1" \
-indicator_reset_1 "diCompReset1" \
-complete_indicator_2 "doPartComp2" \
-incomplete_indicator_2 "doPartIncomp2" \
-indicator_reset_2 "diCompReset2" \
-complete_indicator_3 "doPartComp3" \
-incomplete_indicator_3 "doPartIncomp3" \
-indicator_reset_3 "diCompReset3" \
-complete_indicator_4 "doPartComp4" \
-incomplete_indicator_4 "doPartIncomp4" \
-indicator_reset_4 "diCompReset4"
```

3.5 Production Monitor without Production Manager

Overview

Production Monitor is best used with the Production Manager Execution Engine. However, an interface has been provided to give designers of custom execution engines access to the cycle-specific information in Production Monitor.

The execution engine must tell Production Monitor when a part cycle starts and ends. Plus it must provide certain information about the part cycle that will be recorded in the Production Monitor tables. Two instructions, `ProcCycleStart` and `ProcCycleEnd` are used for this purpose.

Instruction signatures:

```
PROC ProcCycleStart (
num EventID,
num CycleID,
string Part,
string PartDescription,
num PartCount,
num Station)

PROC ProcCycleEnd (
num EventID,
num CycleID,
num PartCount)
```

Example

The custom execution engine should call these instructions before and after executing a part cycle. Below is a basic example of how that would be implemented:

```
%%%
VERSION:1
LANGUAGE:ENGLISH
%%%
MODULE CustomEngine(SYSMODULE)
PROC main()
MyEngine;
ENDPROC
PROC MyEngine()
VAR num pvEventID;
VAR num pvCycleID;
VAR num pvPartCount;
WHILE TRUE DO
WaitUntil <TimeToDoSomething>;
pvEventID:=pvEventID+1;
IF <TimeToRunPart> THEN
pvCycleID:=pvCycleID+1;
! Get all the part info and tell ProcessWare...
ProcCycleStart pvEventID,pvCycleID,
"MyPart","This is my part",pvPartCount,1;
<Execute the part routine>
```

Continues on next page

```
pvPartCount:=pvPartCount+1;
ProcCycleEnd pvEventID, pvCycleID, pvPartCount;
pvCycleID:=0;
ELSEIF <TimeToRunService> THEN
! run the service or setup routine...
ENDIF
ENDWHILE
ENDPROC
ENDMODULE
```

4 RAPID instructions

4.1 GapGetCycleID - Cycle ID accessor for Production Manager

Usage

This instruction is provided by the Production Manager Execution Engine to give access to the current Cycle ID number.

Example

```
GapGetCycleID nID;
```

When called, the variable `nID` is updated with the current CycleID value.

Arguments

```
CycleID
```

CycleID

Data type: `num`

Value to be updated. Must be declared as VAR or PERS.

Program execution

Sequence: When called, the variable supplied in the field, CycleID, is updated with the current CycleID value.

Execution in stepwise mode

Forward: Fully executed.

Backward: Not supported.

Fault management

No handle-able errors.

Syntax

```
GapGetCycleID
[ CycleID ':=' ] < expression (INOUT) of num > ';'
```

Related information

For information about	See
GapGetEventID	*GapGetEventID - Event ID accessor for Production Manager on page 32*.

4.2 GapGetEventID - Event ID accessor for Production Manager

Usage

This instruction is provided by the Production Manager Execution Engine to give access to the current Event ID number.

Example

```
GapGetEventID nID;
```

When called, the variable `nID` is updated with the current EventID value.

Arguments

```
EventID
```

EventID

Data type: `num`

Value to be updated. Must be declared as VAR or PERS.

Program execution

Sequence: When called, the variable supplied in the field, EventID, is updated with the current EventID value.

Execution in stepwise mode

Forward: Fully executed.

Backward: Not supported.

Fault management

No handle-able errors.

Syntax

```
GapGetEventID
[ EventID ':=' ] < expression (INOUT) of num > ';'
```

Related information

For information about	See
GapGetCycleID	*GapGetCycleID - Cycle ID accessor for Production Manager on page 31*.

4.3 GetArcAccum - Get accumulated arc time and count

Usage

This method is used to retrieve the accumulated arc time and arc start count from the Arc Production Monitor.

Example

```
GetArcAccum nArcTime, nArcStarts;
```

The variables `nArcTime` and `nArcStarts` are updated with the current values stored in the Arc Production Monitor.

Arguments

```
ArcTime ArcStarts
```

ArcTime

Data type: `num`

Accumulated arc-on time in seconds.

ArcStarts

Data type: num

Accumulated arc starts.

Program execution

Sequence: This method is used to retrieve the accumulated arc timer and arc start counter from Production Monitor. The values could be used to control automatic torch cleaning frequency, for example.

These values normally grow indefinitely until reset using `ResetArcAccum`.

Execution in stepwise mode

Forward: Fully executed.

Backward: Not supported.

Fault management

No handle-able errors.

Syntax

```
GetArcAccum ';'
[ ArcTime ':='] < expression (INOUT) of num > ','
[ ArcStarts ':=' ] < expression (INOUT) of num > ';'
```

Related information

For information about	See
ResetArcAccum	*ResetArcAccum - Reset accumulated arc time and count on page 40*.
Accumulated Arc Timer	*Accumulated arc time and arc starts on page 23*.

4.4 GetCurrPartCount - Part count setter

Usage

Accessor provided by Production Manager to permit the system logic to retrieve the part count for the current part cycle. The part count is incremented within the Production Manager Execution Engine automatically.

Example

```
GetCurrPartCount nPartCount;
```

The variable `nPartCount` is updated to match the current value stored in Production Manager.

The Production Manager Execution Engine maintains the value internally, only while a part cycle is active. For this reason the instruction `GetCurrPartCount` must be used to retrieve and store the value.

Arguments

```
PartCount
```

PartCount

Data type: `num`

Value to be retrieved from Production Manager.

Detailed example

```
MODULE SystemLogic(SYSMODULE)
CONST menudata mdClearPartCount :=
["Clear Part Counter",
"MyImage.gif",
"ClearPartCount", 3,
"", GAP_SHOW_ALWAYS,
TRUE, GAP_SERVICE_TYPE,
255, FALSE,0];
LOCAL VAR num nPartCount;
PROC BeforePart()
SetCurrPartCount nPartCount;
ENDPROC
PROC AfterPart()
GetCurrPartCount nPartCount;
ENDPROC
PROC ClearPartCount()
nPartCount:=0;
ENDPROC
ENDMODULE
```

Execution in stepwise mode

Forward: Fully executed.

Backward: Not supported.

Continues on next page

Fault management

No handle-able errors.

Syntax

```
GetCurrPartCount
[ PartCount ':='] < expression (INOUT) of num > ';'
```

Related information

For information about	See
SetCurrPartCount	*SetCurrPartCount - Part count setter on page 41*.

4.5 ProcCycleEnd - Process Cycle End

Usage

This instruction is used by system execution engines to inform Production Monitor when a part cycle is finished.

Example

```
ProcCycleStart pvEventID,pvCycleID,
"MyPart","This is my part",pvPartCount,1;
<Execute the part routine>
ProcCycleEnd pvEventID, pvCycleID, pvPartCount+1;
```

Relevant part cycle data is supplied to the Production Monitor by calling `ProcCycleStart` and `ProcCycleEnd`.

Arguments

```
EventID CycleID PartCount
```

EventID

Data type: `num`

This number should increment each time the engine is told to perform an action. It is used as a key when querying Production Monitor data.

CycleID

Data type: `num`

This number should be incremented each time the engine is told to execute a part cycle. Alternatively, the value could be fed from an external source, such as a plant-wide management system. Used as such, the CycleID may be used as a serial number for part tracing. The CycleID is used as a key when querying Production Monitor data.

PartCount

Data type: `num`

The current count to display in the records. The value is not incremented by Production Monitor. It should be incremented prior to calling `ProcCycleEnd`.

Program execution

Sequence: When called data is passed to Production Monitor. Production Monitor reacts by flushing its stored data to the database.

Execution in stepwise mode

Forward: Fully executed.

Backward: Not supported.

Fault management

No handle-able errors.

Syntax

```
PartCycleStart
```

Continues on next page

```
[ EventID ':=' ] < expression (IN) of num > ','
[ CycleID ':=' ] < expression (IN) of num > ','
[ PartCount ':=' ] < persistent (IN) of num > ';'
```

Related information

For information about	See
ProcCycleStart	*ProcCycleStart - Process Cycle Start on page 38*.
Production Monitor without Production Manager	*Production Monitor without Production Manager on page 28*.

4.6 ProcCycleStart - Process Cycle Start

Usage

This instruction is used by system execution engines to inform Production Monitor when a part cycle is about to begin.

Example

```
ProcCycleStart pvEventID,pvCycleID,
"MyPart","This is my part",pvPartCount,1;
<Execute the part routine>
ProcCycleEnd pvEventID, pvCycleID, pvPartCount+1;
```

Relevant part cycle data is supplied to the Production Monitor by calling `ProcCycleStart` and `ProcCycleEnd`.

Arguments

```
EventID CycleID Part PartDescription PartCount
```

EventID

Data type: `num`

This number should increment each time the engine is told to perform an action. It is used as a key when querying Production Monitor data.

CycleID

Data type: `num`

This number should be incremented each time the engine is told to execute a part cycle. Alternatively, the value could be fed from an external source, such as a plant-wide management system. Used as such, the CycleID may be used as a serial number for part tracing. The CycleID is used as a key when querying Production Monitor data.

Part

Data type: `string`

The part name is recorded in the Production Monitor tables. It is also used by Production Monitor when building the file name for reference data storage. This should match the procedure name of the part routine.

PartDescription

Data type: `string`

A description of the part.

PartCount

Data type: `num`

The current count to display in the records. The value is not incremented by Production Monitor.

Continues on next page

Program execution

Sequence: When called data is passed to Production Monitor. Until the `ProcCycleEnd` is called or PP is moved to the main, Production Monitor uses the supplied data in the data records it writes.

Execution in stepwise mode

Forward: Fully executed.

Backward: Not supported.

Fault management

No handle-able errors.

Syntax

```
PartCycleStart
[ EventID ':=' ] < expression (IN) of num > ','
[ CycleID ':=' ] < expression (IN) of num > ','
[ Part ':=' ] < expression (IN) of string > ','
[ PartDescription ':=' ] < expression (IN) of string > ','
[ PartCount ':=' ] < persistent (IN) of num > ';'
```

Related information

For information about	See
ProcCycleEnd	*ProcCycleEnd - Process Cycle End on page 36*.
Production Monitor without Production Manager	*Production Monitor without Production Manager on page 28*.

4.7 ResetArcAccum - Reset accumulated arc time and count

Usage

This method is used to reset the accumulated arc timer and arc start counter in the Production Monitor for Arc.

Example

```
ResetArcAccum;
```

When called, the internal timer and counter for Arc is reset for the task.

Arguments

Data type:

Program execution

Sequence: This method is used to reset the accumulated arc timer and arc start counter in the Production Monitor for Arc. These values normally grow indefinitely until reset using `ResetArcAccum`. The values are recorded in Production Monitor data tables along with other cycle and arc statistics.

Execution in stepwise mode

Forward: Fully executed.

Backward: Not supported.

Fault management

No handle-able errors.

Syntax

```
ResetArcAccum ';'
```

Related information

For information about	See
GetArcAccum	*GetArcAccum - Get accumulated arc time and count on page 33*.
Accumulated Arc Timer	*Accumulated arc time and arc starts on page 23*.

4.8 SetCurrPartCount - Part count setter

Usage

Accessor provided by Production Manager to permit the system logic to update the part count for the current part cycle. The part count is reported in Production Monitor data tables for that cycle.

Example

```
SetCurrPartCount nPartCount;
```

The value stored in `nPartCount` is passed to Production Manager to be available to Production Monitor for data logging.

Arguments

```
PartCount
```

PartCount

Data type: `num`

Value to be displayed by Production Monitor in data logged during that part cycle.

Detailed example

```
MODULE SystemLogic(SYSMODULE)
  CONST menudata mdClearPartCount :=["Clear Part Counter",
        "MyImage.gif", "ClearPartCount", 3, "", GAP_SHOW_ALWAYS,
        TRUE, GAP_SERVICE_TYPE, 255, FALSE,0];
  LOCAL VAR num nPartCount;
  PROC BeforePart()
    SetCurrPartCount nPartCount;
  ENDPROC
  PROC AfterPart()
    GetCurrPartCount nPartCount;
  ENDPROC
  PROC ClearPartCount()
    nPartCount:=0;
  ENDPROC
ENDMODULE
```

Program execution

Sequence: When called the `PartCount` value is supplied to the Production Monitor Execution Engine. As the Execution Engine supplies cycle events to Production Monitor, data is recorded with the supplied Part Count.

Execution in stepwise mode

Forward: Fully executed.

Backward: Not supported.

Fault management

No handle-able errors.

Continues on next page

Syntax

```
SetCurrPartCount
  [ PartCount ':='] < expression (IN) of num > ';'
```

Related information

For information about	See
GetCurrPartCount	*GetCurrPartCount - Part count setter on page 34*.

5 Database tables

5.1 Arc

Seam Events - tblSeamEv

Column name	Data type	Description
EventID	Long Integer	A number supplied by GAP Execution Engine.
CycleID	Long Integer	A number supplied by GAP Execution Engine.
Condition	Integer	Enumeration for Start or End.
Part	String	Path to, and name of, part procedure.
PartDescription	String	From partdata description.
Station	Integer	Current station number.
SeamName	String	Name supplied in SeamName optional argument.
Location	Float	Distance from weld start.
Segment	Integer	Weld segment number.
UserID	String	User that is logged-in.
RobotID	String	Task name.
ControllerID	String	Supplied by SCWrite.
Time	DateTime	Supplied by SCWrite.

Seam Results - tblSeamRes, SeamResults.csv

Column name	Data type	Description
EventID	Long Integer	A number supplied by GAP Execution Engine.
CycleID	Long Integer	A number supplied by GAP Execution Engine.
Part	String	Path to, and name of, part procedure.
PartDescription	String	From partdata description.
Station	Integer	From partdata station.
SeamName	String	Name supplied in Arc instruction.
ArcStartDuration	Number	How long it takes to ignite arc.
NominalArcStart	Number	Nominal length of time to ignite arc.
SeamLen	Number	Length of actual weld completed for the seam.
NominalSeamLen	Number	Saved nominal length of seam.
Duration	Number	Time in seconds to complete seam.
NominalDuration	Number	Saved nominal time in seconds to complete part.
ArcStarts	Integer	Number of arc starts for the seam - ideally 1.
Stops	Integer	Number of stops during welding for any reason.
Quality	Number	Quality in %. This is supplied in *Weld Data Monitoring*.
RequiredQuality	Number	The required quality in %. This is supplied in *Weld Data Monitoring*.

Continues on next page

Column name	Data type	Description
Completed	Boolean	True if all welds finished to completion.
UserID	String	User that is logged in.
RobotID	String	Task name.
ControllerID	String	Supplied by SCWrite.
Time	DateTime	Supplied by SCWrite.

Arc Cycle Results - tblCycleRes, CycleResults.csv

Column name	Data type	Description
EventID	Long Integer	A number supplied by GAP Execution Engine.
CycleID	Long Integer	A number supplied by GAP Execution Engine.
Part	String	Path to, and name of, part procedure.
PartDescription	String	From partdata description.
PartCount	Integer	Count at time of event. A 'Start' condition will show the part count before running the part. An 'End' condition should show an incremented PartCount.
Station	Integer	From partdata station.
WeldLen	Number	Accumulated weld lengths for all welds in part.
NominalWeldLen	Number	Saved accumulated weld lengths for all welds in part.
Duration	Number	Time in seconds to complete part.
NominalDuration	Number	Saved nominal time in seconds to complete part.
Welds	Integer	Number of welds completed during the part cycle.
NominalWelds	Integer	Saved number of welds completed during the part cycle.
ArcStarts	Integer	Number of arc-starts during the cycle.
WeldStops	Integer	Number of stops during welding for any reason.
Completed	Boolean	True if all welds finished to completion.
FaultySeams	String	List of faulty seam index numbers.
AccumArcTime	Number	Total arc-time for the robot.
AccumArcStarts	Number	Total number of arc starts for the robot.
UserID	String	User that is logged in.
RobotID	String	Task name.
ControllerID	String	Supplied by SCWrite.
Time	DateTime	Supplied by SCWrite.

5.2 Production manager

Production Manager Events - tblGapEv, GapEvent.csv

Column name	Data type	Description
EventID	Long Integer	A number supplied by GAP Execution Engine.
EventType	Integer	Enumeration for Cycle, Service, and Setup types.
Condition	Integer	Enumeration for Start or End.
UserID	String	User that is logged in.
RobotID	String	Task name.
ControllerID	String	Supplied by SCWrite.
Time	DateTime	Supplied by SCWrite.

Cycle Events - tblCycleEv, CycleEvent.csv

Column name	Data type	Description
EventID	Long Integer	A number supplied by GAP Execution Engine.
CycleID	Long Integer	A number supplied by GAP Execution Engine.
Condition	Integer	Enumeration for Start or End.
Part	String	Path to, and name of, part procedure.
PartDescription	String	From partdata description.
PartCount	Integer	Count at time of event. A 'Start' condition will show the part count before running the part. An 'End' condition should show an incremented PartCount.
Station	Integer	From partdata station.
UserID	String	User that is logged in.
RobotID	String	Task name.
ControllerID	String	Supplied by SCWrite.
Time	DateTime	Supplied by SCWrite.

5.3 BullsEye

BullsEye Results - tblBullsEye, BullsEye.csv

Column name	Data type	Description
EventID	Long Integer	A number supplied by GAP Execution Engine.
CycleID	Long Integer	A number supplied by GAP Execution Engine.
ToolName	String	Name the tooldata instance.
DispMag	Number	Magnitude of TCP offset change (mm).
NewX	Num	X component of new TCP (mm).
NewY	Num	Y component of new TCP (mm).
NewZ	Num	Z component of new TCP (mm).
NewRotX	Num	Rotational component around X of new TCP (degrees).
NewRotY	Num	Rotational component around Y of new TCP (degrees).
NewRotZ	Num	Rotational component around Z of new TCP (degrees).
DeltaX	Num	Deviation of X component from Day 1 (mm).
DeltaY	Num	Deviation of Y component from Day 1 (mm).
DeltaZ	Num	Deviation of Z component from Day 1 (mm).
DeltaRotX	Num	Rotational deviation of X component from Day 1 (degrees).
DeltaRotY	Num	Rotational deviation of Y component from Day 1 (degrees).
DeltaRotZ	Num	Rotational deviation of Z component from Day 1 (degrees).
Action	String	Name of BullsEye instruction called.
Duration	Number	Time in seconds to complete action.
UserID	String	User that is logged in.
RobotID	String	Task name.
ControllerID	String	Supplied by SCWrite.
Time	DateTime	Supplied by SCWrite.

5.4 Navigator

Search Sphere Results - tblNavSrch, SearchSp.csv

Column name	Data type	Description
EventID	Long Integer	A number supplied by GAP Execution Engine.
CycleID	Long Integer	A number supplied by GAP Execution Engine.
NomX	Num	Nominal X component.
NomY	Num	Nominal Y component.
NomZ	Num	Nominal Z component.
deltaX	Num	Delta X component.
deltay	Num	Delta Y component.
deltaz	Num	Delta Z component.
deltaTot	Num	Magnitude of displacement.
MeanError	Num	Mean Error.
MaxError	Num	Max Error.
SearchName	String	Name derived from the optional argument in the Navigator RAPID instruction interface.
UserID	String	User that is logged in.
RobotID	String	Task name.
ControllerID	String	Supplied by SCWrite.
Time	DateTime	Supplied by SCWrite.

Measure1D - tblNavMeas, Measure1D.csv

Column name	Data type	Description
EventID	Long Integer	A number supplied by GAP Execution Engine.
CycleID	Long Integer	A number supplied by GAP Execution Engine.
NomX	Num	Nominal X component.
NomY	Num	Nominal Y component.
NomZ	Num	Nominal Z component.
MeasureName	String	Name derived from the optional argument in the Navigator RAPID instruction interface.
deltaTot	Num	Magnitude of displacement.
UserID	String	User that is logged in.
RobotID	String	Task name.
ControllerID	String	Supplied by SCWrite.
Time	DateTime	Supplied by SCWrite.

5.5 SmarTac

Search1D - tblSmtc1D, Search1D.csv

Column name	Data type	Description
EventID	Long Integer	A number supplied by GAP Execution Engine.
CycleID	Long Integer	A number supplied by GAP Execution Engine.
SearchName	String	Name derived from the optional argument in the SmarTac™ RAPID instruction interface.
DispMag	Num	Magnitude of displacement.
DispSpec	Num	Maximum allowed displacement.
ResultX	Num	The search result value after completion represented as a frame.
ResultY	Num	The search result value after completion represented as a frame.
ResultZ	Num	The search result value after completion represented as a frame.
ErrorType	Num	Number indicating possible errors during action.
Duration	Num	Time in seconds to complete search.
UserID	String	Logged-in user name.
RobotID	String	Task name.
ControllerID	String	Supplied by SCWrite.
Time	DateTime	Supplied by SCWrite.

SearchPart - tblSmtcPart, SearchPart.csv

Column name	Data type	Description
EventID	Long Integer	A number supplied by GAP Execution Engine.
CycleID	Long Integer	A number supplied by GAP Execution Engine.
SearchName	String	Name derived from the optional argument in the SmarTac™ RAPID instruction interface.
Detected	Bool	True if contact with part is made.
ErrorType	Num	Number indicating possible errors during action.
Duration	Num	Time in seconds to complete search.
UserID	String	Logged-in user name.
RobotID	String	Task name.
ControllerID	String	Supplied by SCWrite.
Time	DateTime	Supplied by SCWrite.

SearchGroove - tblSmtcGrv, SearchGroove.csv

Column name	Data type	Description
EventID	Long Integer	A number supplied by GAP Execution Engine.
CycleID	Long Integer	A number supplied by GAP Execution Engine.

Continues on next page

Column name	Data type	Description
SearchName	String	Name derived from the optional argument in the SmarTac™ RAPID instruction interface.
ResultX	Num	The search result value after completion represented as a frame.
ResultY	Num	The search result value after completion represented as a frame.
ResultZ	Num	The search result value after completion represented as a frame.
GrooveWidth	Num	Width of actual groove in mm.
NomWidth	Num	Nominal groove width expected in mm.
ErrorType	Num	Number indicating possible errors during action.
Duration	Num	Time in seconds to complete search.
UserID	String	Logged-in user name.
RobotID	String	Task name.
ControllerID	String	Supplied by SCWrite.
Time	DateTime	Supplied by SCWrite.

5.6 Torch service center

Torch Clean Results - tblTchClean, TchService.csv

Column name	Data type	Description
EventID	Long Integer	A number supplied by GAP Execution Engine.
CycleID	Long Integer	A number supplied by GAP Execution Engine.
Action	String	Name of the TorchClean instruction called.
ToolName	String	Name of the tooldata instance used in the Torch-Clean instruction.
UserID	String	Logged-in user name.
RobotID	String	Task name.
ControllerID	String	Supplied by SCWrite.
Time	DateTime	Supplied by SCWrite.

Index

Contact us

ABB AB
Discrete Automation and Motion
Robotics
S-721 68 VÄSTERÅS, Sweden
Telephone +46 (0) 21 344 400

ABB AS, Robotics
Discrete Automation and Motion
Nordlysvegen 7, N-4340 BRYNE, Norway
Box 265, N-4349 BRYNE, Norway
Telephone: +47 51489000

ABB Engineering (Shanghai) Ltd.
5 Lane 369, ChuangYe Road
KangQiao Town, PuDong District
SHANGHAI 201319, China
Telephone: +86 21 6105 6666

www.abb.com/robotics

3HAC050963-001, Rev -, en

Power and productivity
for a better world™