

# ***PGCKW and Offline Configuration***

## ***TPS V5.0 Network***

PGCKW and Offline Configuration

Issue: 1.00

TPS V5.0 Network 04/03/98

Copyright Micro SciTech Ltd. 1991-1998

Information in this document is subject to change without notice and does not represent a commitment on the part of Micro SciTech Ltd. The software described in this document is furnished under a licence agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the licence or nondisclosure agreement. No part of this manual may be reproduced or transmitted in any form or by means electronic or mechanical, including photocopying and recording, for any purpose without the express permission of Micro SciTech.

Copyright Micro SciTech Ltd., 1991-1998. All rights reserved.

TPS and TPS M4 is a trademark of Micro SciTech Ltd.

Windows, MS-DOS, Windows NT, Windows for Workgroups and Windows 95 are registered trademarks of Microsoft Corporation.

This page is intentionally left blank.

1	Disclaimer .....	4
2	Configuration Overview.....	5
	2.1 Page Display Files .....	6
	2.2 Procedure Overview.....	7
	2.3 Using PGCKW .....	7
	2.4 Windows 3.1/95/NT Compatibility Issues.....	9
	2.5 Porting (upgrading) Display Pages .....	9
3	Display Page Language .....	10
	3.1 Block specification .....	11
	3.2 Identifier Summary.....	13
	3.3 Online/Offline Nomenclature .....	16
	3.4 Minimum View-port Specification .....	19
	3.5 Minimum Plot-port Specification.....	20
	3.6 Identifier Descriptions.....	23
4	Page files as a Parameter Database .....	53
5	Changing the System Pages .....	56
6	PGCKW Warnings and Errors.....	57
	6.1 Warnings.....	58
	6.2 Errors .....	62
7	Displayrate and DCWs.....	67

## 1 Disclaimer

**Users making offline configuration changes to the text pages do so entirely at their own risk. Micro SciTech cannot accept any liability for failure of TPS when using pages modified offline.**

It was the original intention of Micro SciTech to phase-out PGCKW and put all its functions online. (PGCKW was originally used for TPS DOS which didn't have online configuration). However, following the release of TPS M4, version 4.0, it was found to be very useful and has now been left in the product for versions 4.0, 4.1 and 5.0. It will probably (very likely) be left in for future versions.

However, as PGCKW was originally to be deleted, development work was stopped early on in the TPS M4 for Windows development programme (before V4.0 release) and, consequently, PGCKW does not perform many very important checks performed online when editing a view-port's settings. Unfortunately, it is possible to enter illegal values which will pass through the compiler unnoticed but may crash TPS when viewing, hence the disclaimer:

Nevertheless, if you do have problems and cannot resolve them, Micro SciTech will endeavour to resolve the problems via Technical Support.

The safest way to make offline edits is to modify existing attributes rather than add new attributes although this is not a strict limitation and attributes may be added if users feel confident.

## 2 Configuration Overview

Offline configuration is the process of configuring TPS display pages from outside of TPS using a separate text editor and the TPS display page compiler PGCKW. TPS stores the display configuration information in text files which can be modified offline with a standard text editor. Offline configuration allows you to perform all the standard online configuration operations, such as changing a view-port's colors or its position, plus some more powerful editing features not currently available online.

For instance, offline configuration is superior to online configuration for the following operations

- Global page changes of view-ports attributes, e.g. changing all view-port's background color

- Adding/deleting whole groups of parameters quickly

- Adding/deleting parameters from telemetry parameter database files

- Including pre-defined pages

- Defining a view-port to appear on every page

Offline Configuration is still a necessity for the following operations which cannot be performed online.

- Changing the parameter record 'tag parameter'

- changing the system view-port attributes (color changes only)

Offline editing is best used for changing existing attributes rather than creating new ports or adding attributes. Creating a new port is always best done online and performed by copying an existing port rather than creating a new port (which is, ultimately, just a copy of a template port pre-defined on page 9998). However, adding a port by simply cutting it from another page file (or any text file for that matter) is a good quick method of adding ports and brings with it the benefits of a **parameter database** concept. See section 4.0 for more details.

## 2.1 Page Display Files

All page display configuration information, as read directly by TPS, is stored in binary files PAGEnnnn.BIN where nnnn is the page number (Automatically backed-up page files have the .BAK extension).

Additionally, every time a page is saved online, TPS calls the page compiler 'PGCKW' to reverse engineer the binary file PAGEnnnn.BIN to build the text readable version PAGEnnnn.TXT. The '.TXT' file is that which can be edited offline. When used offline, PGCKW is used to compile the text file and build the binary file '.BIN' which is then read by TPS when you change to the page.

When the TPS backup is enabled, TPS will regularly save a backup of the current page configuration in the binary file PAGEnnnn.BAK. If you do not save your page, accidentally or otherwise, this backup file can be used in place of the .BIN file - rename it to PAGEnnnn.BIN first.

If you inspect a page file such as PAGE0010.TXT using a standard text editor, you can see the language TPS uses to describe a view-ports attributes - it is relatively intuitive and it will only take you a matter of minutes to see the correlation between the online configuration and the information stored in the text files.

### To summarise

Page display information is stored in up to three separate files

PAGEnnnn.BIN	(that which TPS reads and writes to online)
PAGEnnnn.TXT	(Generated from PAGEnnnn.BIN by PGCKW when saving a page)
PAGEnnnn.BAK	(that which TPS automatically generates online when 'Backup' is enabled)

PAGEnnnn.BIN and PAGEnnnn.TXT are only ever generated when a page is saved (Display, Save menu item).

PAGEnnnn.TXT can be offline edited and recompiled to build PAGEnnnn.BIN - a binary file readable by TPS.

PAGEnnnn.BAK is automatically generated every 29 seconds (the default backup time period), see the TPS Options, System, Backup menu item.



## 2.2 Procedure Overview

Offline configuration comprises the following steps

1. Edit the display page file PAGEEnnnn.TXT and modify attributes as desired.
2. Compile the text page PAGEEnnnn.TXT to a TPS readable file binary file PAGEEnnnn.BIN, using PGCKW
3. View the binary file PAGEEnnnn.BIN using TPS M4

To edit the text page file, you need to understand the TPS page description language. This is described in section 3.

## 2.3 Using PGCKW

PGCKW is a basic Windows program which performs either one of two functions.

1. compile PAGEEnnnn.TXT to build PAGEEnnnn.BIN
2. reverse engineer PAGEEnnnn.BIN to retrieve PAGEEnnnn.TXT

PGCKW is not automatically setup as an icon in the TPS for Windows group. The reasons for this are historical and primarily due to the originally envisaged deletion of PGCKW from the TPS product. Instead, PGCKW is run from the program manager via the File, Run menu item as for the TPS setup program.

Note, although PGCKW has scroll bars, these are not functional by design.

### Compilation Example

To compile page 15 (PAGE0015.TXT) and build PAGE0015.BIN

1. Run the TPS page compiler, PGCKW as follows:

select Start, Run from the Windows 95/NT4 Start menu or File, Run from the Windows 3.1/NT3.51 Program Manager menu, then enter C:\TPSV5\PGCKW - this assumes you have used the default installation of TPS to the C drive directory TPSV5, adjust the text otherwise.

From the PGCKW window, compile page 15 by selecting the File, Compile menu item and then enter 15 (or select PAGE0015.TXT from the list box). PGCKW will then compile the page and, if all is ok, you will see the following message:

```
Processing page file \PAGE0015.TXT. Output to file PAGE0015.LST
All error messages echoed to the output file and display
PGCK: All page files checked ok.
```

If there are any problems in the compilation, you will see blue warning messages and/or red error message. Blue warning messages are harmless and you will be able to view the compiled page with TPS. One or more red error messages imply the compilation failed and you will not be able to view the page (PGCKW will not create a binary file PAGEnnnn.BIN). Warning messages and errors are documented in section 6 of this manual.

2. Exit PGCKW - select the `File, Exit` menu-item or press ALT Q as for quitting TPS.
3. Start TPS and turn to page 15 to verify the conversion.

### Reverse-Engineering Example

To reverse engineer page 15 (PAGE0015.BIN) to build PAGE0015.TXT

1. Run the TPS page compiler, PGCKW as follows:

select `Start, Run` from the Windows 95/NT4 Start menu or `File, Run` from the Windows 3.1/NT3.51 Program Manager menu, then enter C:\TPSV5\PGCKW - this assumes you have used the default installation of TPS to the C drive directory TPSV5, adjust the text otherwise.

From the PGCKW window, reverse engineer page 15 by selecting the `File, Reverse` menu item and then enter 15 (or select PAGE0015.TXT from the list box). If the file PAGE0015.TXT already exists, PGCKW will prompt you for confirmation before overwriting any existing text page file. if you press Yes to overwrite, PGCKW will then reverse engineer the page and you will see the following message:

```
Reverse engineering file \PAGE0015.BIN. Output to file PAGE0015.TXT
All error messages echoed to the output file and display
PGCK: All page files checked ok.
```

If you simply recompiled a text page without making any offline modifications, you should not see any errors since the compiled page is, by definition, error free. If you do receive errors, it implies the binary page file is corrupt. In which case, try using the backup version.

2. Exit PGCKW - select the `File, Exit` menu-item or press ALT Q as for quitting TPS.
3. View the text page PAGE0015.TXT.

## 2.4 Windows 3.1/95/NT Compatibility Issues

The compiled page display files PAGEEnnnn.BIN are compatible across all TPS for Windows models., i.e. a TPS for Winows 3.1 model can directly read, without any recompilation, pages compiled for TPS for Windows NT/95. The converse is also true.

Whilst the compiled and source page files are compatible, PGCKW still comes in a separate Windows 3.1 version and NT/95 version. The only reason for this is that PGCKW for Windows NT/95 is a true 32-bit version whilst PGCKW for Windows 3.1 is 16-bit. The PGCKW for Windows NT/95 is therefore slightly faster although you will probably not notice.

## 2.5 Porting (upgrading) Display Pages

Porting display pages is the procedure to convert a display page from an older to a newer version of TPS. It may also be possible to convert from a newer to an older version of TPS although this cannot be guaranteed.

NOTE:

**porting == upgrading == converting == compiling a page**

Porting is simply a matter of recompiling the PAGEEnnnn.TXT file using the PGCKW utility supplied with the TPS version to which you are upgrading. Do not use an older version of PGCKW.

For example, to convert page 99 from TPS version 4.1 (or 4.0) to TPS version 5.0, you would recompile page 99 using the PGCKW executable supplied with V5.0.

### 3 Display Page Language

The attributes of all view-ports is specified offline using a simple specification language similar to that used for Windows '.INI' files, i.e. it takes the form:

identifier = value

The best way to familiarise yourself with the language is to look at a page file. Below is an extract for the 'PKT\_MENMONIC' view-port as found on page 9998 (file PAGE9998.TXT) and used as a template when creating new view-ports (double click an empty part of the screen and click the New button). Note, an alarm was configured to fill out the example - the alarm does not appear on page 9998.

```
[valueport]
  databasekey = "PKT_MNEMONIC"
  view = packetheader
  byteindex = 2
  datatype = "%6hd"
  row = 10
  column = 3
  mnemonic = "PKT_MNEMONIC"
  title = "PKT_TITLE"
  units = "PKT_UNITS"
  displayrate = "F16ST03"
  backgroundcolor = blue
  bordercolor = white
  titlecolor = white
  valuecolor = yellow
  border = off
[calibration]
  function = polynomial
  [polynomial]
    bitmask = 0x0000ffff
    a4 = 0
    a3 = 0
    a2 = 0
    a1 = 1
    a0 = 0
[alarm]
  blink = on
  sound = on
  stepmode = on
  invert = on
  filename = "C:\TPSW31\pkt_mnem.txt"
  lowerlimit = 0
  upperlimit = 10
```

The display specification for a view-port takes the form of a block of lines starting with the **[valueport]** header line. The term 'valueport' is a DOS remnant and has been superceded in TPS M4 by the term 'view-port'. However, it remains as 'valueport' for backward compatibility. All view-port specification is then indented with separate sub-blocks, e.g. [calibration] and [alarm] as shown above.

**For 'view-port', read [valueport]**

### 3.1 Block specification

Each parameter's view-port attributes starts with the **[valueport]** header line. All attributes that follow are indented below and to the right of the [valueport] and the attributes are specified in the following form:

identifier = value

**IMPORTANT:** The space either side of the '=' sign is important and should be retained.

The [valueport] specification block is not formally terminated and TPS only determines the block's end when it encounters the next [valueport] or [system] block header. This is contrary to some programming languages which have a 'BEGIN' and 'END' block marker.

The identifier can be, for instance, `backgroundcolor`, with a value such as `green`. The specification would then take the form:

```
backgroundcolor = green
```

All specification is case insensitive and all upper case letters are converted to lower case.

The block specification scheme for [valueport] is repeated at sub-block levels. A good example of this is the [plotport] block specification which is actually a sub-block of the [valueport]

Thus, a pagefile [valueport] could have a layout as shown below:

```
[valueport]
  value-port specification block line 1
  value-port specification block line 2
  .
  .
  [plotport]
    plot port specification block line 1
    plot port specification block line 2
    .
    .
```

It is important to left-align all identifiers within the same block, e.g., the below specification

```
[valueport]
  view = datapacket
  bytearray = 15
  row = 10
```

has the identifiers `view` and `byteindex` correctly aligned but the `row` identifier has been accidentally indented one space and so will be incorrectly interpreted by TPS as a new sub-block.

It is simplest to use a tab for indentation, each nested level being indented one tab further to the right than the level in which it is embedded. Always ensure you use consistent tab settings. Failure to indent consistently can generate many errors when using PGCKW.

**For consistency and compatibility with other applications we strongly recommend using the good old DOS default of 1 tab = 8 spaces.**

There are other sub-blocks in addition to [plotport], namely [alarm], [filter] and [calibration], the latter also has its own sub-blocks: [bitoperation], [polynomial]

Within a specification block, the general order of the identifiers is immaterial albeit we advise you retain the sequence set by PGCKW when it reverse engineers the text file. This makes it easier for future compatibility and can help eliminate possible problems in offline configuration.

For example, the following specification:

```
[valueport]
.
.
row = 10
column = 15
.

; End of value-port specification block
```

is identical to:

```
[valueport]
.
.
column = 15
row = 10
.

; End of value-port specification block
```

Where the row and column identifiers have been swapped in sequence.

An important case where the order must be retained, is the use of the **rawformat**, **displayformat** and **datatype** identifiers which must come before any [polynomial], [bitoperation], [alarm] or [filter] block specification. This is because TPS needs the rawformat, datatype and displayformat information to process the polynomial or bitmask values appearing in the [polynomial], [bitoperation], [alarm] or [filter] specification blocks.

## 3.2 Identifier Summary

A summary description of identifiers found in the text page files is tabulated below:

Because the offline configuration was primarily developed for the older DOS TPS, there are several cases where the nomenclature doesn't match with the TPS for Windows dialog box nomenclature. A summary of all identifiers and their online equivalents is tabulated in the next section following the below identifier summary descriptions.

The valueport identifiers are listed below in the order in which they appear in a page file.

Identifier	Description
[valueport]	view-port
databasekey	unique view-port name automatically extracted from the first 20 characters of the mnemonic, title or units.
rawformat	online view-port 'Datatype', i.e. raw format
view	the parameter displayed in the view-port.
byteindex	byte offset of view-port value from the start of the parameter. Oomitted if zero.
supercommcount	number of super-commutated samples in the packet, omitted if single sampled.
supercommincr	byte offset between consecutive super-commutated samples, omitted if single sampled
datatype	raw data-type as found in the packet. Identical to the rawformat identifier.
displayformat	display format of the view-port parameter value, omitted if same as rawformat ('Datatype').
row	view-port bottom row position.
column	view-port left column position.
mnemonic	parameter mnemonic. Omitted if blank.
title	view-port title, omitted if blank.
units	parameter units, omitted if blank.
displayrate	encoded version of the parameter refresh rate.
backgroundcolor	view-port background color.
bordercolor	view-port border color.
titlecolor	view-port title color.
valuecolor	parameter value color.
filename	parameter record filename, omitted if not recorded.
sbttagcount	sub-com tag parameter count (RESERVED), omitted if not referenced as a sub-com tag parameter.

[calibration]		calibration sub-block, omitted if no calibration function selected.
function		calibration function
[polynomial]		polynomial calibration sub-block.
modulus		polynomial calibration modulus.
quotient		polynomial calibration quotient.
function		polynomial byteswap or reciprocal function.
bitmask		polynomial calibration bit-mask.
a4,a3...a0		polynomial calibration coefficients $((a4 \times x + a3) \times x + a2) \times x + a1) \times x + a0$ .
avgsamples		number of samples averaged
[bitoperation]		calibration bit-test function sub-block, omitted if not a bit-test operation, e.g. 'onoffbit'.
bitmask		calibration bit-test mask.
[alarm]		alarm sub-block, omitted if no alarm set.
blink	alarm Blink	switch.
sound		alarm Beep switch.
stepmode		alarm Breakpoint switch.
invert	alarm Invert	switch.
filename		alarm Record filename.
lowerlimit		alarm lower limit.
upperlimit		alarm upper limit.
[filter]		packet filter sub-block, omitted if no filter configured.
lowerlimit		filter lower limit.
upperlimit		filter upper limit.
invert	filter Invert	switch
[subcom]		sub-commutation sub-block, omitted if single-sampled.
subcomtag		sub-com tag parameter databasekey.
cvtype		sub-com calibration data type (RESERVED).
portno		sub-com tag view-port number (RESERVED).
lowerlimit		sub-com tag lower limit.
upperlimit		sub-com tag upper limit.



[plotport]	plot-port sub-block, omitted if no plot configured.
row	plot-port bottom row position.
column	plot-port left column position.
width	width of plot in characters.
height	height of plot in characters.
xtitle	X axis horizontal title, omitted if blank.
ytitle	Y axis vertical title, omitted if blank.
title	main plot title, omitted if blank.
axiscolor	axes title colors.
axislabelcolor	axes, scale factor and gradation color.
backgroundcolor	plot area background color.
bordercolor	no online equivalent (RESERVED).
gridcolor	plot grid color.
plotcolor	plot trace color.
titlecolor	plot main title title
xlower	X Lower limit.
ylower	Y Lower limit.
xupper	X Upper limit.
yupper	Y Upper limit.
maxpoints	maximum points on a plot
autoplotx	Auto Plot X, omitted if not enabled.
autoploty	Auto Plot Y, omitted if not enabled.
autorefreshx	Auto Refresh X, omitted if not enabled.
autorefreshy	Auto Refresh Y, omitted if not enabled.
lineplot	Line-plot, omitted if not enabled.
plotgrid	overlaid grid, omitted if not enabled.
forwardplot	forward.trace plot switch
backwardplot	backward trace plot switch

### 3.3 Online/Offline Nomenclature

A summary of all offline identifiers and their online dialog box equivalents is tabulated in this section.

The identifiers are listed in the order in which they appear in a page file.

Some online edit items (see right-hand column) are ambiguous, i.e. they appear twice in the same dialog box. For instance the **Title** identifier is found in both the view-port settings 'Text' and 'Colors' group. Where this is the case, the edit group is enclosed in brackets on the right.

The Offline identifiers, given in the below left-hand column, are always unique within a sub-block but some identifiers, such as **lowerlimit**, can appear in more than one sub-block. For instance, in the case of 'lowerlimit', this is found within the [alarm], [filter] and [subcom] sub-blocks. In such cases, the offline identifiers all have the same definition and are not ambiguous.

Offline Identifier	Online Edit item
[valueport]	view-port settings dialog box
databasekey	no online equivalent (RESERVED)
view	Parameter
byteindex	Byteindex
supercommcount	Count
supercommincr	Increment
rawformat	Datatype
displayformat	Format
datatype	Datatype, Format (when the same)
row	Row
column	Column
mnemonic	Mnemonic
title	Title (Text group)
units	Units
displayrate	Rate
backgroundcolor	Background
bordercolor	Border (Colors group)
titlecolor	Title (Colors group)
valuecolor	Value
border	Border (Switches group)
drawenable	No draw
filename	Parameter record file dialog box
sbttagcount	no online equivalent (RESERVED)

[calibration] function	polynomial or bit-operation dialog box Function
[polynomial] modulus	polynomial dialog box Modulus
quotient	Quotient
function	Byteswap or reciprocal check-box
bitmask	Bitmask
a4,a3...a0	Coefficients A4, A3 ... A0
avgsamples	Average count
[bitoperation] bitmask	bit-test operation, e.g. onoffbit dialog box Bitmask
[alarm] blink	Alarm dialog box Blink
sound	Beep
stepmode	Breakpoint
invert	Invert
filename	Record-to-file dialog box
lowerlimit	Lower limit
upperlimit	Upper Limit
[filter] lowerlimit	Transfer, Filter pkts menu item dialog box Lower limit
upperlimit	Upper limit
invert	Invert
[subcom] subcomtag	Subcom dialog box sub-com tag parameter dialog box
cvtype	no online equivalent (RESERVED)
portno	no online equivalent (RESERVED)
lowerlimit	Lower limit
upperlimit	Upper limit

[plotport]	Plot settings dialog box
row	Row
column	Column
width	Width
height	Height
xtitle	X Title
ytitle	Y Title
title	Plot Title
axiscolor	Axis Titles
axislabelcolor	Axis Labels
backgroundcolor	Background
bordercolor	no online equivalent (RESERVED)
gridcolor	Grid
plotcolor	Plot
titlecolor	Title (Colors group)
xlower	X Lower
ylower	Y Lower
xupper	X Upper
yupper	Y Upper
maxpoints	Max points
autoplotx	Auto Plot X
autoploty	Auto Plot Y
autorefreshx	Auto Refresh X
autorefreshy	Auto Refresh Y
lineplot	Line-plot
plotgrid	Grid (Switches group)
forwardplot	Forward
backwardplot	Backward

### 3.4 Minimum View-port Specification

Whilst there are many identifiers, a view-port specification can be extremely minimal, the following is an example of the [valueport] minimum specification required to display a packet parameter. (The prefixed semi-colon is used for comments).

```
; This is a minimal view-port specification
; It displays a 2-byte short integer at byteindex 10 in the packet
; The view-port
[valueport]
; no 'view = ' statement implies packet parameter
byteindex = 10; zero bias offset from 'packetheader'
datatype = "%hu" ; 2-byte short integer
row = 10 ; bottom left row 10, column 30
column = 30
mnemonic = "minimum" ; could use title or units identifiers instead
```

If no **view** identifier is present, it is assumed the view-port references the packetheader.

Whilst the **mnemonic** identifier is shown, a single line with the **title** or **units** identifier could be used in its place. The specification is complete so long as at least one of the three identifiers is present with a non-blank value.

The default settings for the identifiers not present are as follows. Note, PGCKW will not generate many of these lines when reverse engineering a page.

#### [valueport] assumed defaults for a minimum view-port

```
databasekey = "minimum" ; 1st 20 chars of mnemonic
view = packetheader ; packet parameter view-port
supercommcount = 1 ; single sampled
supercommincr = 0 ; ditto
rawformat = "%hu" ; same as datatype
displayformat = %hu% ; same as datatype
title = "" ; blank, no title
units = "" ; blank, no units
displayrate = "every10msecs" ; fastest display rate
backgroundcolor = blue ; default colors
bordercolor = white
titlecolor = white
valuecolor = yellow
border = on ; default border enabled
drawenable = off ; would be text only view-port if off
filename = "" ; no parameter recording
sbttagcount = 0 ; not a sub-com tag port
```

The [plotport], [subcom], [alarm], [calibration], [filter] sub-blocks are not present as TPS assumes the following for a minimum specification:

- the view-port has no plot-port
- the view-port is not a sub-commutated parameter
- the view-port has no alarm, calibration or filter configured.

Note, super-commutation and parameter recording, specified via the **supercommcount-****supercommincr** and **filename** identifiers respectively are specified within the [valueport] block and do not have a separate [] sub-block.

### 3.5 Minimum Plot-port Specification

A minimum specification for a plot-port is shown below. It is quite a bit more extensive than a single minimum view-port. This is mainly because any plotted parameter (the Y parameter), must be preceded by an X value view-port. Additionally, each X and Y view-port must have a polynomial calibration function to convert from raw (usually integer) to floating point format and the Y view-port must have an extra [plotport] sub-block to specify the plot-port attributes.

See the next page...

```

; This is a minimal plot-port specification
; It plots a 2-byte short integer at byteindex 10 in the packet
; the X axis view-port is defined first. This is a 2-byte short
; integer at byteindex 2.
[valueport]                                ; X parameter, view-port
    byteindex = 2
    rawformat = "%hu"                      ; raw format in packet
    displayformat = "%f"                   ; convert to float for plot
    row = 20                               ; bottom left row 10, column 30
    column = 20
    mnemonic = "xminimum"                  ; could use title or units identifiers instead
    [calibration]                          ; required to convert to float for plot
        function = polynomial
        [polynomial]
            a4 = 0.
            a3 = 0.
            a2 = 0.
            a1 = 1.
            a0 = 0.
[valueport]                                ; Y parameter, view-port
    byteindex = 10
    rawformat = "%hu"
    displayformat = "%f"                   ; convert to float for plot-port
    row = 20                               ; bottom left row 10, column 30
    column = 10
    mnemonic = "yminimum"                  ; could use title or units identifiers instead
    [calibration]                          ; required to convert to float for plot
        function = polynomial
        [polynomial]
            a4 = 0.
            a3 = 0.
            a2 = 0.
            a1 = 1.
            a0 = 0.
[plotport]
    row = 16                               ; 4 rows above Y view-port
    column = 10
    xlower = 0.
    ylower = 0.
    xupper = 30.
    yupper = 100.

```

Comparing with the single minum view-port...

It is seen that **datatype** is now split into **rawformat** and **displayformat** which is as for online configuration. The datatype is only used when rawformat and displayformat are the same - since the raw integer value must be converted to a floating point for a plot, the two separate formats are now required. To do the conversion, a simple linear polynomial function also has to be added, hence the extra **[calibration]** and **[polynomial]** sub-blocks.

The additional **[plotport]** sub-block is relatively minimal. The only mandatory identifiers used are the plot's row and column position which denote the bottom left of the entire plot frame (includes all the axes labels). Thereafter, only the axis limits need specifying.

The default plot size, specified by the **width** and **height** identifiers do not need specifying, nor do any axis titles. PGCKW assumes the following [plotport] default specification:

```
width = 38      ; 38 character columns plot width
height = 11     ; 11 rows plot height
xtitle = ""     ; blank
ytitle = ""     ; blank
title = ""      ; blank
axiscolor = white ; X, Y axis titles - default blank
axislabelcolor = white ; See note below
backgroundcolor = blue ; visible plot area blue
bordercolor      ; no online equivalent (RESERVED)
gridcolor = white ; info only - grid not enabled by default
plotcolor = yellow ; pixel trace color - lineplot off
titlecolor = white ; info only - main plot title blank
autoplotx = off   ; absolute plot limits, no auto-scaling
autoploty = off   ; ditto
autorefreshx = off ; no refresh when trace is outside limits
autorefreshy = off ; ditto
lineplot = off    ; pixel plot only
plotgrid = off    ; no overlaid grid
forwardplot = off ; points can plot to left or right, i.e. scatter
backwardplot = off ; points can plot up or down, i.e. scatter
maxpoints = 1024  ; 1024 unique pixels stored per plot
```

The **axislabelcolor** is actually the color of the X and Y scale factors and gradation markings. This is white by default which unfortunately means they are not visible with the TPS default white window background (see the Options, System, Colors menu item). This is an error stemming from the DOS days when the default background window color was black.

The default plot-port has absolute axis limits, i.e. no autoplot axis scaling and no autorefresh when the plot trace is outside the plot limits. There is no overlaid grid, neither are there any titles.



## 3.6 Identifier Descriptions

This section contains a detailed description of each of the offline identifiers used in the page specification language. The identifiers are listed in the approximate order in which they appear in the text page file.

### Databasekey

RESERVED FOR SYSTEM USE - INFORMATION ONLY

This is an internally derived attribute generated by PGCKW when compiling.

The databasekey has no equivalent in the TPS view-port settings dialog box

If you modify this, PGCKW will unconditionally overwrite it when next reverse engineering the page.

The databasekey is intended as a unique identifier to the view-port parameter.

**More often than not, the databasekey is synonymous with the parameter 'mnemonic'**

When TPS displays any parameter list-box, the names listed are actually the database keys for all view-ports on the page referenced. For example, when using the 'Copy from' TPS option, TPS extracts the databasekeys for all parameters in the source-page (that page from where the parameter is to be copied) and displays them in a drop-down list-box.

PGCKW derives the database key as

```
if (the mnemonic is non-blank) then databasekey = mnemonic
else if (the title field is non-blank) then databasekey = title
else if (the units field is non-blank) then databasekey = units
else {
    // set default values
    set databasekey = "XXXX"
    set mnemonic = "XXXX"
}
```

Thus, entering a non-blank entry for all three is illegal as it would imply the view-port has no identifiable databasekey. However, PGCKW will not reject such a view-port but will, instead, give the view-port a default value of "XXXX" for the mnemonic and databasekey. PGCKW will issue the following warning when such view-ports are encountered.

```
Parameter name defaulted to XXXX
```

Of course, if several such cases are found, the XXXX mnemonic will not be unique. This is harmless but undesirable. You can edit the view-port settings online to change all such XXXX occurrences.

The databasekey is always embedded in double quotes.

### Example

Using the page 9998 PKT\_MNEMONIC example, it is seen that the databasekey has been equated with the mnemonic as shown below.

```
databasekey = "PKT_MNEMONIC"
.
mnemonic = "PKT_MNEMONIC"
title = "PKT_TITLE"
units = "PKT UNITS"
```

Note, although advisable, it is not mandatory for the databasekey to be unique and TPS does not perform uniqueness checks.

Only the first 20 characters of the mnemonic, title or units are used for the databasekey.

TPS will extract only the significant characters up to the first delimiter which can be a space, comma or tab.

**view**

Synonymous with the equivalent view-port settings 'Parameter' list box.

View-ports displaying any packet parameters always use the specification:

$\text{view} = \text{packetheader}$

Since users are primarily only interested in packet parameters, the **view** identifier is generally not modified by a user. Otherwise, see the online help topic 'Parameter'.

**byteindex**

Synonymous with the equivalent view-port settings **Byteindex** edit item.

This identifier will not appear if the online setting is 0. When viewing a packet parameter, i.e. 'view = packetheader', a byteindex of zero points to the first sync byte. Normally, packet parameter view-ports will have a non-zero byteindex and, consequently, the byteindex identifier will be present in the [valueport] block.

When viewing packet parameters, this can have any value from 0 to 4223 where 4224 is the maximum packet size in bytes.

**WARNING - If you exceed the byte length of the view-port parameter, you could cause TPS to crash.** The results are unpredictable. TPS performs an online check to ensure this does not happen but **this check is NOT currently performed offline by PGCKW.**

Because the **supercommcount** and **supercommincr** (see below) can add to the byteindex, having the byteindex attribute less than the parameter (packet) length does not guarantee that there are no errors.

$\text{actual byteindex} = \text{byteindex} + \text{supercommcount} \times \text{supercommincr}$

See the online help 'Super-commutation' topic.

**supercommcount**

The number of samples in the packet

Synonymous with the equivalent view-port settings **Count** edit item.

This identifier will not appear if the online setting is 1, i.e. the parameter is single sampled.

Used in combination with the **byteindex** and **supercommincr** identifiers. See the **byteindex** for more details.

**supercommincr**

The byte offset between consecutive super-commutated samples.

Synonymous with the equivalent view-port settings **Increment** edit item.

This identifier will not appear if the online setting is 0, i.e. the parameter is single sampled.

Used in combination with the **byteindex** and **supercommcount** identifiers. See the **byteindex** for more details.

**sbtagcount****RESERVED FOR SYSTEM USE - INFORMATION ONLY**

An abbreviation for 'sub-com-tag counter'

This is an internally derived attribute generated by TPS when loading a page. It is actually a count of the number of sub-commutated view-port parameters that reference this sub-com-tag view-port.

If a view-port is not a sub-com-tag port, PGCKW will not generate this line.

Note, a sub-com tag view-port is not the same as a sub-commutated view-port. When you online configure a sub-commutated view-port, you are requested to select a sub-com tag parameter. This sub-com tag parameter is used by TPS to determine when the sub-commutated parameter is to be extracted from the packet (via numeric limits as for an alarm or filter). A sub-commutated view-port has its own separate **[subcom]** block (see further in this section) whereas a sub-com tag parameter only has a 'sbtagcount =' line and does not possess a **[subcom]** block.

sbtg\_count has no equivalent in the TPS view-port settings dialog box.

If you modify this, PGCKW will unconditionally overwrite its value when next reverse- engineering the page.

See the Samples PAGE0009.TXT for an example of this identifier.

A sub-com parameter, i.e. that with a [subcom] block, cannot also have a sbtagcount entry.

If a view-port is not a sub-com-tag port, PGCKW will not generate this line.

### **rawformat**

A parameter's raw format as found in the packet.

Synonymous with the online view-port settings **Datatype** edit item.

This is also synonymous with the offline identifier '**datatype**' (lower case d) when both the online **Datatype** and display **Format** edit items are identical (see further below).

The value can be any C language printf format string, enclosed in double quotes, or a TPS simplified datatype, e.g.

rawformat = "%hu"	; C language short unsigned 2-byte integer
rawformat = "real"	; Simplified single precision 4-byte floating point no
rawformat = "%16b"	; 8 bit binary integer (not standard C format)

PGCKW will generate a rawformat identifier line if the **rawformat** is NOT identical to the **displayformat**. Otherwise, PGCKW will generate a single **datatype** identifier.

The datatype can be any valid C format or simplified data-type.

See also the online help topics 'datatype' and 'C printf'.

## **datatype**

A parameter's **rawformat** and **displayformat** data-type when both are the same.

For instance, the following specification:

```
rawformat = "%hu"           ; raw format 2 byte unsigned short integer
displayformat = "%hu"       ; display as 2 byte unsigned short integer
```

is identical to, and shortened by PGCKW to:

```
datatype = "%hu"
```

Although having the same name as the online view-port settings **Datatype** edit item (upper case 'D'), it is only synonymous when the offline **rawformat** and **displayformat** are identical.

PGCKW will generate this line if the **rawformat** is identical to the **displayformat**. Otherwise, this identifier is not used and PGCKW generates a separate line for **rawformat** and **displayformat**.

The datatype can be any valid C format or simplified data-type. See also the **rawformat** identifier.

The datatype value must always be enclosed in double quotes.

## **displayformat**

The display format of the view-port parameter value.

Synonymous with the online view-port settings **Format** edit item.

Any valid C format or simplified data-type

PGCKW will not generate a displayformat entry if the **rawformat** and **displayformat** are identical. Instead, it will generate a single **datatype** specification line

The **displayformat** value must always be enclosed in double quotes.

See also the online help topics 'datatype' and 'C printf' formats

**row**

View-port bottom row position

Synonymous with the equivalent view-port settings **Row** edit item.

Range 1-80

Best modified online by dragging and dropping the view-port

See the online help topic 'datatype' and 'C Printf' formats

**column**

view-port left column position

Synonymous with the equivalent view-port settings **Column** edit item.

Range 1-160

Best modified online by dragging and dropping the view-port

**Mnemonic, Title and Units**

Synonymous with the equivalent view-port settings **Mnemonic, Title, Units** dialog box items.

The options must be enclosed in string double quotes.

A blank value is the same as no entry. For instance, **units** = "" denotes an empty units field and, consequently, the next time the page is reverse-engineered, a units specification will not appear.

There must always be at least one non-blank entry for the Mnemonic, Title or Units.

See also the **databasekey** identifier for more information.

**displayrate**

An encoded version of the parameter refresh rate.

Synonymous with the equivalent view-port settings **Rate** edit item.

The encoded value is of the format 'FnnSTmm' (where nn and mm are both two hex digits) and is called a Display Control Word 'DCW'. The encoded DCW is a DOS remnant and will probably be phased out in future versions of TPS and replaced with the standard 'everynnmsecs' form. For those interested, there is a section on DCWS at the end of this manual.

Although PGCKW always writes an encoded form, you can use the online rate specification, ie.

`displayrate = "everynnmsecs"`

where nn can be 0 to 320. PGCKW will automatically adjust it to one of the following, 10, 20, 40, 80, 160 or 320 msecs. 320 msecs is the maximum period when the loop-time is 10 msecs, i.e. the default upon load-up.

To interpret the encoded forms generated by PGCKW, the FnnSTmm is converted to a rate as follows:

<b>online 'Rate'</b>		<b>offline DCW 'FnnSTmm'</b>	
every10msecs	==	F01ST01	
every20msecs	==	F02STmm	mm = 00 or 01
every40msecs	==	F04STmm	mm = 00 to 03
every80msecs	==	F08STmm	mm = 00 to 07
every160msecs	==	F16STmm	mm = 00 to 15
every320msecs	==	F32STmm	mm = 00 to 31

mm is arbitrary and internally computed by TPS. For information only.

All plot parameters, i.e. those [valueport] blocks that have an embedded [plotport] sub-block must always be set to F01ST01 as follows:

`displayrate = "F01ST01"` ; equivalent to "every10msecs"

This also applies to those [valueport] blocks with an embedded [alarm], [record] or [filter] sub-block which always run at the fastest rate.

See also section 7.0 of this manual.



**backgroundcolor, bordercolor, titlecolor, valuecolor**

Synonymous with the view-port edit items shown in the **Colors** edit group

backgroundcolor	view-port background color
bordercolor	view-port border color
titlecolor	view-port title color
valuecolor	parameter value color

Each of the four view-port colors can be anyone of the sixteen colors exactly as found in the view-port settings drop-down list boxes.

The 16 possible colors are:

black, blue, brown, cyan, darkgrey, green, highblue, highcyan, highgreen, highmagenta, highred, lightgrey, magenta, red, white, yellow.

Note, the color values are NOT embedded in double quotes.

If these colors are omitted, TPS will default to the following:

```
backgroundcolor = blue
bordercolor = white
titlecolor = white
valuecolor = yellow
```

Hence the abundance of blue background view-ports in early TPS page displays!

Note, unlike other defaults whereby PGCKW does not generate a line in the text file, all four color attributes always have a line generated even when the colors are the default values.

See also the Samples disk pages 42-64 for example view-ports with all the different color combinations.

**border**

Synonymous with the equivalent view-port settings **Border** check box found within the **Switches** group.

If the valueport **Border** option is NOT enabled, PGCKW will generate the statement

```
border = off
```

If the valueport **Border** option is enabled, PGCKW will not generate a line.

Note, in offline configuration, the border is default on unless explicitly disabled. Thus, if you omit the line 'border = off', PGCKW will compile a view-port with a border.

The border uses the old DOS box drawing characters which add 2 characters to the height and width.

**drawenable**

Synonymous with the equivalent view-port settings **No draw** check box found within the **Switches** group.

The drawenable switch is commonly disabled when creating text-only view-ports for the creation of title or comment text on a page.

If the valueport **No draw** option is enabled, PGCKW will generate the statement

```
drawenable = off
```

If the online view-port settings **No draw** check-box is disabled (i.e. unchecked), PGCKW will not generate a line.

**[calibration] sub-block**

The [calibration] sub-block contains all the attributes as assigned online when using any Function other than dummy\_calibfn, dummy\_calibfn1 or dummy\_calibfn2 which switch the calibration process off. When off, PGCKW does not generate a [calibration] sub-block.

**See the online help topic 'Calibration' for full details on all calibration aspects.**

**Example**

The page 9998 PKT\_MNEMONIC calibration sub-block has a polynomial function configured with a specification as follows:

```
[calibration]
  function = polynomial
  [polynomial]
    bitmask = 0x0000ffff
    a4 = 0
    a3 = 0
    a2 = 0
    a1 = 1
    a0 = 0
```

## function

This specifies a calibration function, as found in the online view-port settings **Function** list box.

The function identifier must always be present within a **[calibration]** sub-block.

Commonly, the function value is **polynomial** which, when present, has its own sub-block **[polynomial]** containing the polynomial attributes such as the coefficients etc. as shown above. If the function is a mathematical function such as **sin**, it too will be followed by a polynomial sub-block.

Another common function is **onoffbit** which, when present, has its own sub-block **[bitoperation]** containing the single bitmask as shown below:

```
[calibration]
    function = onoffbit
    [bitoperation]
        bitmask = 0x00001000                ; test bit 12
```

A full list of all functions can be found in the online help topic 'Calibration'.

If the function is neither **polynomial**, a bit test function, or a mathematical function, no further sub-blocks are generated. A common example of this is when the **function = hexchartext** which enables the display of packet byte dumps (page 10). This simply has a **[calibration]** sub-block as follows:

```
[calibration]
    function = hexchartext
```

**[polynomial]**

This is a sub-block of the **[calibration]** sub-block, added when the **polynomial** or mathematical calibration function, such as **sin**, is used.

The [polynomial] sub-block contains all those attributes assigned online in the Polynomial Calibration dialog box. It is rare when all attributes are set, however, an example when this could happen is shown below:

```
[calibration]
  function = polynomial
  [polynomial]
    function = longbyteswap           ; 4 byte swap
    modulus = 1.024000e+003
    quotient = 5.120000e+002
    function = reciprocal
    bitmask = 0xff0000ff
    a4 = 0.000000e+000
    a3 = 0.000000e+000
    a2 = 0.000000e+000
    a1 = 1.000000e+000
    a0 = 0.000000e+000
    avgsamples = 10
```

Note, rather confusingly, the order in which the attributes are shown, is not that in which they are performed. Although harmless, this will be rectified in a future release. Please keep in mind the actual order TPS performs the operations is:

```
byteswap
bitmask
modulus
quotient
reciprocal
polynomial
avgsamples
```

You may enter them in the logical order and compile the text pages as such. When PGCKW reverse engineers the page, it will revert the order back to that shown in the above example.

If the polynomial has the online **Byteswap** check-box marked as enabled, PGCKW generates one of the following three lines:

```
function = integerbyteswap           ; for 2 byte Datatype, e.g. "%hu"
function = longbyteswap ; for 4 byte Datatypes, e.g. "%lu"
function = doublebyteswap           ; for 8 byte Datatype, e.g. "%lf"
```

TPS automatically determines which function to use by examining the view-port settings **Datatype** (not the display **Format**) and substitutes in the appropriate function. Thus, offline, there is no direct equivalent of the online **Byteswap** on/off setting.

### Bitmask

This identifier is only present if the online polynomial settings **Bitmask** edit item is non-blank. An online blank entry is effectively treated as a full mask 0xFFFFFFFF in which case all bits of the raw uncalibrated value are left unchanged.

Omit the bitmask line if no bit-mask is to be applied. Adding a bitmask = 0xFFFFFFFF is harmless but will consume a bit more CPU time (won't be noticeable!).

Bitmasks can be entered in hexadecimal, binary octal or decimal with the preceeding '0x', '0b', '0o' and '0d' type identifier.

They are always reverse engineered as full 32-bit masks irrespective of whether the raw value is less than 32-bits. However, it is harmless to enter shorter masks e.g. for a 2-byte short integer. (TPS internally, and transiently, converts raw integer data-types to 32-bits when applying a polynomial).

If you have specified a `bitmask` in a page file without using a hex specifier or other type specifier e.g.

```
[bitoperation]
    bitmask = 00100000           ; 100,000 decimal
```

TPS will assume that it is in decimal. Therefore, it is preferable to prefix a 0x hex specifier so that the above example will read:

```
[bitoperation]
    bitmask = 0x00100000         ; 100,000 hexadecimal
```

**modulus**

This identifier is only present if the online polynomial settings **Modulus** edit item is non-blank.

Omit the modulus identifier if no modulus is to be applied.

The modulus value must NOT be zero.

The modulus should be entered in the same format as the online view-port settings display **Format** edit item. For example, if the **displayformat** is of integer type, enter the **modulus** as integer and NOT in real format. The converse case of entering an integer value (i.e. omitting a decimal point) for a real display format is acceptable.

**quotient**

This identifier is only present if the online polynomial settings **Quotient** edit item is non-blank.

Omit the quotient identifier if no quotient is to be applied.

The quotient value must NOT be zero.

The quotient should be entered in the same format as the online view-port settings display **Format** edit item. For example, if the **displayformat** is of integer type, enter the **modulus** as integer and NOT real format. The converse case of entering an integer value, i.e. omitting a decimal point, for a real display format is acceptable.

**reciprocal**

This identifier is only present if the online polynomial settings **Reciprocal** check box is marked. In such a case, the below line is generated by PGCKW.

function = reciprocal

**a4, a3, a2, a1 ,a0 Coefficients**

PGCKW generates a line for each of the coefficients including the case when they are zero.

The coefficients should be entered in the same format as the online view-port settings display **Format** edit item.

**filename**

The parameter record filename of a view-port with parameter recording enabled.

This identifier is found both within the **[valueport]** block and within the **[alarm]** sub-block.

When a filename is present in the **[valueport]** block it specifies that the parameter is being recorded.

Similarly, when a filename is present in the **[alarm]** sub-block, it specifies that the parameter is being recorded.

The value, embedded in double quotes, specifies the full path with drive and directory name, e.g.  
filename = "c:\tpsv5\lastpktr.txt"

**[alarm]**

The **[alarm]** block is a sub-block of a value-port and specifies the alarm attributes of a parameter.

An **[alarm]** sub-block is only generated if the view-port has an online configured alarm.

**blink**

If the alarm **Blink** option is enabled, PGCKW will generate the statement

```
blink = on
```

If the alarm **Blink** option is disabled, PGCKW will not generate a line

**sound**

If the alarm **Beep** option is enabled, PGCKW will generate the statement

```
sound = on
```

If the alarm **Beep** option is disabled, PGCKW will not generate a line



**stepmode**

If the alarm **Breakpoint** option is enabled, PGCKW will generate the statement

stepmode = on

If the alarm **Breakpoint** option is disabled, PGCKW will not generate a line

**invert**

If the alarm **Invert** option is enabled, PGCKW will generate the statement

invert = on

If the alarm **Invert** option is disabled, PGCKW will not generate a line

**filename**

If the alarm **Record** option is on, PGCKW will generate a statement of the form

filename = *filepath*

where *filepath* is the full pathname of the file the parameter is recorded to when the parameter goes out of range of the alarm limits.

The filename specification is identical in format to that used for **[valueport]** parameter record files to which you are referred for more format details.

**lowerlimit, upperlimit**

These two identifiers are common to the **[alarm]** and **[filter]** sub-blocks and always appear in each of the respective sub-blocks when an alarm or a filter is configured.

The identifiers specify the lower and upper limits of the alarm or filter as entered online in the **Upper** and **Lower** alarm/filter edit boxes.

Do not confuse these identifiers with the **[plotport]** identifiers **xlower, xupper, ylower** and **yupper**.

The limits should be entered in the same format as the online view-port settings display **Format** edit item.

NOTE - there are no quotes around the limits even if the limits are text such as 'ON' shown in the example below.

**Examples**

For a fixed or floating point display format parameter, e.g. Format "%f" or "%e" with alarm limits of 5.1 and 6.5 respectively, PGCKW would generate the following text:

```
lowerlimit = 5.1  
upperlimit = 6.5
```

For a view-port with a bit-test calibration function **onoffbit**, an alarm could be set with identical lower and upper limits, ON and ON respectively such that when the value flips to OFF, an alarm is raised. In such a case, PGCKW would generate the following text

```
lower = ON      ; no quotes on limits  
upper = ON
```

**[filter]**

The [filter] block is a sub-block of a value-port and specifies the packet filter attributes associated with a parameter.

A [filter] sub-block is only generated if the view-port has an associated filter.

There can only be one filter on a single page. Whilst, offline, you can add several filters, TPS will only process the first view-port in a page file with a filter set.

A filter is similar to an alarm in its config and uses only a sub-set of its configuration identifiers, namely,

```
[filter]
    lowerlimit
    upperlimit
    invert
```

Specification of these attributes is identical to that of the [alarm] sub-block.

**Example**

The below [valueport] for the PKTSRD parameter (Packets read) shows a [filter] configured such that TPS filters only those incoming packets when the count of incoming packets is outside the range 100-200 inclusive. The 'outside range' is achieved by setting the 'invert = on'. Without this, TPS would filter packets when the count is between 100 and 200 inclusive.

```
[valueport]
    databasekey = "PKTSRD"
    view = vtelfr_cnt
    rawformat = "%11ld"
    displayformat = "%10lu"
    row = 12
    column = 2
    mnemonic = "PKTSRD  "
    displayrate = "F08ST08"
    backgroundcolor = lightgrey
    bordercolor = white
    titlecolor = black
    valuecolor = yellow
    border = off
    [filter]
        lowerlimit = 100
        upperlimit = 200
        invert = on
```

See the Getting Started manual for more details on filtering packets.

**[subcom]**

The subcom block is a sub-block of the [valueport] block.

If a parameter is sub-commutated, PGCKW will generate a [subcom] block in the page file otherwise no subcom block will be present.

See samples PAGE0009.TXT for an example of this identifier.

**subcomtag**

This is the **databasekey** of the sub-com-tag parameter referenced by this sub-commutated view-port. Somewhere in the same page file will be a view-port with the subcomtag value as its **databasekey** (that view-port will have a **sbtgacount** identifier denoting that it is referenced by this [subcom] block).

The general specification structure of a sub-com parameter and its associated sub-com tag parameter is illustrated below. Note, the tag [valueport] is shown first - it needn't necessarily come before the sub-com parameter [valueport] as TPS will scan the entire page for sub-com tags when displaying the page.

```
[valueport]
; Sub-com tag parameter reference by next view-port
databasekey = "subcomtagparam"
.
.
sbtgacount = 1
[valueport]
; sub-commutated parameter referencing sub-com tag
; in previous view-port
databasekey = "subcomparam"
.
[subcom]
subcomtag = "subcomtagparam"
.
```

**cvtype****RESERVED FOR SYSTEM USE - INFORMATION ONLY**

This is an encoded version of the calibration data type of the sub-com-tag parameter. TPS needs this information to correctly interpret the **lowerlimit** and **upperlimit** found in the same [subcom] block, see below.

Do not modify this value.

**portno****RESERVED FOR SYSTEM USE - INFORMATION ONLY**

This is the current view-port number (entry index in TPS internal tables) of the sub-com-tag parameter referenced by this view-port. The portno is internally computed by TPS when loading a page

Do not modify this value.

**lowerlimit, upperlimit**

The limits between which the sub-com-tag parameter must lie so as to extract the sub-commutated parameter from the current packet.

The limits are identical in specification and format rules as for the **[alarm]** sub-block to which you are referred.

**[plotport]**

The [plotport] sub-block specifies all attributes associated with a plot-port.

It is a sub-block of the [valueport] block.

All [plotport] blocks are attached to the Y plotted parameter. The X [valueport] block must always come immediately prior to the Y [valueport] block and TPS will always interpret the preceeding view-port as an X [valueport]. Note, an X valueport specification has no special identifiers declaring it to be such, it is inferred simply by preceeding the Y valueport.

**row**

Plot-port bottom row position

Synonymous with the equivalent plot-port settings **Row** edit item.

Range 1-80

Best modified online by dragging and dropping the plot area.

**column**

Plot-port left column position

Synonymous with the equivalent plot-port settings **Column** edit item.

Range 1-160

Best modified online by dragging and dropping the plot area.

**width**

Width of the plot area, including all axis labels, in characters.

Synonymous with the equivalent plot-port settings **Width** edit item.

The visible plot width is reduced by a maximum of 6 to accommodate up to 6 columns to the left of the Y axis for gradation markings, the Y scale factor and title.

**height**

Height of the plot area, including all horizontal axis labels, in characters.

Synonymous with the equivalent plot-port settings **Height** edit item.

The visible plot area is reduced by a maximum of 3 to accommodate up to 3 rows below the X axis for the gradation markings, the X scale factor and title.

**title**

The main plot title appearing at the top of the plot area.

Synonymous with the equivalent plot-port settings **Plot Title** edit item.

If the title is blank, PGCKW will not generate this line.

**xtitle**

The X axis plot title appearing below the plot area.

Synonymous with the equivalent plot-port settings **X Title** edit item.

If the X title is blank, PGCKW will not generate this line.

**ytitle**

The vertical Y axis plot title appearing to the left of the plot area.

Synonymous with the equivalent plot-port settings **Y Title** edit item.

If the title is blank, PGCKW will not generate this line.

**axiscolor**

Synonymous with the equivalent plot-port settings **Axis Titles** (Colors group) list box.

This is the color of the X and Y axis titles as specified by the **xtitle** and **ytitle** identifiers.

The color can be any one of the 16 colors as shown in the online, plot-port settings list- box. The color range is the same as the [valueport] colors, namely,

black, blue, brown, cyan, darkgrey, green, highblue, highcyan, highgreen, highmagenta, highred, lightgrey, magenta, red, white, yellow.

Note, the color values are NOT embedded in double quotes.

If the colors are omitted, TPS will default to the following:

```
axiscolor = white
axislabelcolor = white ; see note below
backgroundcolor = blue
bordercolor = white
gridcolor = lightgrey
plotcolor = yellow
titlecolor = white
```

The **axislabelcolor** is actually the color of the X and Y scale factors and gradation markings. This is white by default which, unfortunately, means they are not visible with the TPS default white window background (see the Options, System, Colors menu item). This is an error stemming from the DOS days when the default background window color was black.

Unlike other defaults whereby PGCKW does not generate a line in the text file, all color attributes always have a line generated even when the colors are the default values.

### **axislabelcolor**

Synonymous with the online plot-port settings **Axis Labels** (Colors group).

This is the color of the axes, border, scale factor and axes gradations.

See the **axiscolor** (above) for more information.

### **backgroundcolor**

The background color of the visible plot area.

Synonymous with the online plot-port settings **Background** (Colors group).

See the **axiscolor** (above) for more information.

### **bordercolor**

No online equivalent (RESERVED)

The plot border is actually assigned the same color as the **axislabelcolor**, see above.

See the **axiscolor** (above) for more information.

### **gridcolor**

Synonymous with the online plot-port settings **Grid** (Colors group).

See the **axiscolor** (above) for more information.



**plotcolor**

The color of the plot trace

Synonymous with the online plot-port settings **Plot** (Colors group).

See the **axiscolor** (above) for more information.

**titlecolor**

The color of the main plot title

Synonymous with the online plot-port settings **Plot** (Colors group).

See the **axiscolor** (above) for more information.

**xlower**

The lower limit of the X axis.

Synonymous with the online plot-port settings **X Lower** edit item.

Do not confuse this limit with the **[alarm]** and **[filter] lowerlimit** identifier.

All limits can be entered in integer, real, fixed or floating point format, e.g.

123  
123.456  
1.23456e2

TPS automatically adjusts all limits to within 10-20% of the dynamic range of the lower and upper limits when displaying the plots.

For example, limits entered as

xlower = 0  
xupper = 95

are adjusted to

xlower = 0  
xupper = 100

PGCKW reverse engineers the adjusted limits and not the original user entered limits so that the original limits are overwritten in the text page file when the page is next saved online.

### **ylower**

The lower limit of the Y axis.

Synonymous with the online plot-port settings **Y Lower** edit item.

Do not confuse this limit with the **[alarm]** and **[filter] lowerlimit** identifier.

See **xlower** (above) for more details.

### **xupper**

The upper limit of the X axis.

Synonymous with the online plot-port settings **Y Upper** edit item.

Do not confuse this limit with the **[alarm]** and **[filter] upperlimit** identifier.

See **xlower** (above) for more details.

### **yupper**

The upper limit of the Y axis.

Synonymous with the online plot-port settings **Y Upper** edit item.

Do not confuse this limit with the **[alarm]** and **[filter] upperlimit** identifier.

See **xlower** (above) for more details.

**autoplot**

When enabled, TPS rescales both the X and Y plot axes when the plot point is outside the plot limits.

If both autoplotx and autoploty are enabled, PGCKW generates a single line:

```
autoplot = on
```

which is identical to the two individual statements:

```
autoplotx = on
```

```
autoploty = on
```

If only one of the online options **Auto Plot X** and **Auto Plot Y** is enabled, PGCKW generates an individual line for the that option.

There is no directly equivalent online dialog box edit item for **autoplot**, PGCKW determines the autoplot state according to the **Auto Plot X**, **Auto Plot Y** edit items.

**autoplotx**

When enabled, TPS rescales the X plot axis when the plot point is outside the **xlower** or **xupper** plot limit .

Synonymous with the online **Auto Plot X** edit item.

If the online **Auto Plot X** is enabled, but the **Auto Plot Y** is disabled, PGCKW generates a single line

```
autoplotx = on
```

See also the **autoplot** identifier, above.

**autoploty**

When enabled, TPS rescales the Y plot axis when the plot point is outside the **ylower** or **yupper** plot limit .

Synonymous with the online **Auto Plot Y** edit item.

If the online **Auto Plot Y** is enabled, but the **Auto Plot X** is disabled, PGCKW generates a single line

```
autoploty = on
```

See also the **autoplot** identifier, above.

**autorefresh**

When enabled, TPS refreshes the plot trace (blanks it out and restarts) when the plot point is outside either the X or Y plot limits. However TPS does not rescale the axis limits and keeps the same lower and upper limits (unlike **autoplot** which does rescale the axes). It is assumed that the plot values will eventually come back within the range of the existing limits when autorefresh is used.

If both **autorefreshx** and **autorefreshy** are enabled, PGCKW generates a single line:

```
autorefresh = on
```

which is identical to the two individual statements:

```
autorefreshx = on
```

```
autorefreshy = on
```

If only one of the online options **autorefreshx** and **autorefreshy** is enabled, PGCKW generates an individual line that option.

There is no direct dialog box edit item for **autorefresh**, PGCKW determines the autoplot state according to the **Auto Refresh X**, **Auto Refresh Y** edit items.

**autorefreshx**

When enabled, TPS refreshes the plot trace (blanks it out and restarts) when the plot point is outside the X plot limit **xlower** or **xupper**.

Synonymous with the online **Auto Refresh X** edit item.

If the online **Auto Refresh X** is enabled, but the **Auto Refresh Y** is disabled, PGCKW generates a single line

autorefreshx = on

See also the **autorefresh** identifier, above.

**autorefreshy**

When enabled, TPS refreshes the plot trace (blanks it out and restarts) when the plot point is outside the Y plot limit **ylower** or **yupper**.

Synonymous with the online **Auto Refresh X** edit item.

If the online **Auto Refresh Y** is enabled, but the **Auto Refresh Y** is disabled, PGCKW generates a single line

autorefreshy = on

See also the **autorefresh** identifier, above.

**lineplot**

When enabled, straight lines are drawn between consecutive plot-points, otherwise, the plot trace comprises discrete pixel points.

Synonymous with the online plot-port settings **Line-plot** check box

If the plot-port **lineplot** option is enabled, PGCKW will generate the statement

lineplot = on

If the online plot-port **lineplot** option is disabled, PGCKW will not generate a line

**plotgrid**

When enabled, TPS overlays the visible plot area with a grid (graticule).

Synonymous with the online plot-port **Grid** (Switches group) check-box.

If the online plot-port **Grid** option is enabled, PGCKW will generate the statement

```
plotgrid = on
```

If the **Grid** option is disabled, PGCKW will not generate a line.

**forwardplot**

When enabled, TPS will only plot points that trace to the right, i.e. the x value continually increases.

Synonymous with the online plot-port settings, **Forward** check-box.

If the online plot-port **forwardplot** option is enabled, PGCKW will generate the statement

```
forwardplot = on
```

If the **forwardplot** option is disabled, PGCKW will not generate a line.

**backwardplot**

When enabled, TPS will only plot points that trace to the left, i.e. the x value continually decreases.

Synonymous with the online plot-port settings, **Backward** check-box.

If the online **backwardplot** option is enabled, PGCKW will generate the statement

```
backwardplot = on
```

If the **backwardplot** option is disabled, PGCKW will not generate a line.

## 4 Page files as a Parameter Database

This section is provided as a temporary collection of notes. It is the intention to replace TPS page numbers with aliases so that TPS can access any filename which will render much of these notes superfluous.

TPS and PGCKW only work with page files with root names of type PAGEnnnn. However, you can, of course, store the text contents of the PAGEnnnn.TXT files in any file - referred to as the database file. When you wish to build a page, you cut and paste the desired view-ports from the file to the text page file and then compile the page file as usual. The downside of using an arbitrary database filename, e.g. MYFILE.TXT is that TPS and PGCKW cannot directly read or process it and therefore it requires offline manual editing. This can actually be advantageous in the later stages of the project when all view-ports are defined and subject only to minor, irregular modifications. Since TPS cannot access the database, it cannot be accidentally modified.

There are a couple of useful features, namely the ability to include other files from within a file (as in the C language '#include' directive) and the ability to embed comments. Both are detailed further below. Because of limitations in PGCKW, these two features are best used when the database file is not a standard TPS page filename so as to avoid TPS overwriting it.

### **Making the database file a single page file PAGEnnnn.TXT**

If you have less than 1008 view-ports (12 are reserved for page 0, 9999 and 4 are reserved for TPS online workspace), then it is easiest to put all parameters on a single page file. If you have more than 1008 view-ports then you will, of course, have to use two or more pages. Putting 1008 parameters on a page will, make it rather crowded although it is the primary intention that the page is used as a database rather than as a standard mission display page - the database page serving as a source of parameters for inclusion in other, project pages.

Because the database file is a TPS page, as opposed to a standalone text file of arbitrary name, e.g. MYFILE.TXT, it can be viewed and edited online as for any other page. However, it will be overwritten when the page is saved which may or may not be desirable.

## Adding Comments

Comments can be added to text files by preceeding one of the following characters with a semi-colon.

```
; this is a comment  
# this is also a comment  
' so it this
```

### **WARNING: PGCKW does not reverse engineer comments:**

When TPS calls PGCKW to build the text page file, it will not reverse engineer the comment statements This is a good reason to use databases with filenames other than PAGEnnnn.TXT.

## Inclusion of files

An 'include' style statement (actually just the character '@') within a page file can pull in parameters defined within other files, either PAGEnnnn.TXT files or other database files. It can actually pull in any text, not just one or more complete view-ports, i.e. a partial block specification such as a polynomial calibration sub-block [polynomial].

For example, suppose you have a single parameter view-port configured in a file ONEPARM.TXT, then you can pull it into page 10 by including the following line

```
@oneparm.txt
```

This can be placed in column one, left justified, outside or inside any [valueport] block. If the include file contains complete view-ports, i.e. a complete [valueport] specification blocks, then it must come outside of any [valueport] block, i.e. the first statement before and after just before a new [valueport].

### **WARNING: PGCKW does not reverse engineer an include statement:**

When TPS calls PGCKW to build the text page file, it will not reverse engineer the include statement but automatically embed all view-ports as referenced by the include statement. Therefore, the include statement is lost! Beware of this. This is another good reason to use databases with filenames other than PAGEnnnn.TXT.



It may well be a good idea to allocate a separate file (database) for every parameter. You can then build up a page file only by referencing the individual database files. For instance, suppose there were only two parameters stored in files PARM1.TXT and PARM2.TXT. Then, to include both in a TPS page 99, your page 99 file would look as follows:

```
; PAGE0099.TXT
;
@PARM1.TXT      ; parameter 1
@PARM2.TXT      ; parameter 2
```

Because each parameter file contains a single parameter, each would have a single [valueport] specification block (and any sub-blocks). Furthermore, the individual files can also contain include statements.

Two possible parameter files PARM1.TXT and PARM2.TXT are shown below. Note, a minimal specification for each view-port is shown - this is not mandatory, but just to keep the example files short. The second file PARM2.TXT, also includes a [calibration] sub-block defined in file PARM2CAL.TXT.

```
; PARM1.TXT
; minimal specification
[valueport]
  byteindex = 10
  datatype = "%hu"
  row = 10
  column = 30
  mnemonic = "PARM1"

; PARM2.TXT
; minimal specification
[valueport]
  byteindex = 12
  datatype = "%f"
  row = 10
  column = 50
  mnemonic = "PARM2"
@PARM2CAL.TXT      ; include a calibration block

; PARM2CAL.TXT
; '@include'd into PARM2.TXT
; remember to keep all indent levels consistent
; calibration block
[calibration]
  function = polynomial
  [polynomial]
    a1 = 1.0
    a0 = 0.0
```

## 5 Changing the System Pages

The system page 0 contains all the TPS configuration information such as synchronisation bit pattern and packet length. Additionally, it contains all the system view-ports, i.e. those view-ports comprising the blue background displays seen on rows 32-35, columns 1-80. Also see the online help topic 'System Display'.

Because the system display appears on every page, it is evident that any view-port configured on this page 'appears' on every page albeit defined only once in page 0. From a user-perspective, this is a very useful property and an additional system page 9999 is now explicitly reserved for the user to serve this purpose - any user-defined view-port on page 9999 will appear on every page.

By design, pages 0 and 9999 cannot be modified online. Pages 0 and 9999 are therefore modified offline using the standard offline configuration methods detailed throughout this manual.

### Page 0 reserved - do not modify except...

Because TPS modifies page 0 and overwrites it when saving, it is not the intention that you modify it except for the purposes of changing the parameter recording tag parameter - see the online help topic 'tag - changing'. Page 0 is effectively considered reserved and the special identifiers used for holding system configuration information are not documented because of this.

### Page 9999 free for use

Page 9999, as shipped, holds two view-ports. The first view-port is reserved and used as TPS internal workspace - please do not delete it. The second view-port is the TPS message bar (see the online help topic 'Message bar') which can be deleted as for any offline configured view-port, i.e. just delete the entire [valueport] specification block. You are free to add any extra view-ports after the first view-port. All view-port configuration is exactly as for any other user page (1-9900) with the added bonus that **any view-port defined on page 9999 appears on every page**, i.e. is permanently displayed.

## 6 PGCKW Warnings and Errors

This section details all warning and error messages generated by PGCKW. The messages are given in alphabetic order.

As for all language compilers, PGCKW issues warning and error messages. Warning messages, displayed in blue, alert the user to potential problems but are not considered serious and do not stop the generation of a compiled page. Error messages, displayed in red, are serious errors which will stop the generation of a page file so that it cannot be later displayed by TPS. PGCKW will usually take corrective action in the event of a warning to save any potentially hazardous configuration.

### General Points on Page Compiler Errors

Note, one error encountered may spawn a host of related errors. If you are completely mystified as to some errors, try clearing the errors you do understand and then recompile - this may eliminate other difficult errors. For instance, the following specification has an unrecognisable view parameter, namely `zpage_no` which should actually be `page_no`.

```
[valueport]
    title = "Example"
    view = zpage_no
    row = 10
    column = 1
    datatype = "integer"
```

This generates the following compiler output:

```
!! 1 Error Page 99 line 00003 !!Parametername invalid/missing parameter!
!! 2 Error Page 99 line 00003 !!Unable to use line. Identifier unrecognised
!! 3 Error Page 99 line 00013 !!No byte index into packet specified

!! PGCK: encountered 0 warnings and 3 errors !!
```

The 2nd and 3rd errors are entirely due to the first error which arises because `zpage_no` was not valid. Note, the 3rd error arises because, without a valid **view** identifier value, the page compiler takes the default as `packet_header` and then expects a mandatory **byteindex** identifier which isn't present.

Most errors always generate the second error

```
Unable to use line. Identifier unrecognised
```

Some bad lines in a specification block are not detected until the next block is parsed. This means the error is displayed out of position, several lines after the line to which it pertains.

Bad indentation is a common source of errors. Each sub-block must be indented one space or more to the right of its parent block or sub-block. Usually sub-blocks are indented one tab space which is often equivalent to 8 spaces. If just one identifier is not indented to the same level as all other identifiers within a sub-block, PGCKW will generate many errors. Sometimes, when viewing the file, identifiers may appear correctly indented when, in fact, they aren't. This can sometimes occur when using different editors with different tab settings. If in doubt, view the file and temporarily change the tab settings say, from 8 to 5. This can often reveal offending lines.

## 6.1 Warnings

### adjusting row position to within range

The value of the row identifier is outside the legal range of 1-80 rows.

Corrective action:

PGCKW sets the **row** to its maximum value of 80.

### Alarm lower limit missing

The **lowerlimit** identifier has not been specified in an alarm sub-block. The specification of the limits is mandatory.

Corrective action:

PGCKW completely disables the alarm.

### Alarm upper limit missing

The **upperlimit** identifier has not been specified in an alarm sub-block. The specification of the limits is mandatory.

Alarm not created: all options off

All the identifiers blink, sound, stepmode, invert are either disabled or missing, e.g.

blink = off  
sound = off  
stepmode = off  
invert = off

At least one identifier must be present and enabled 'on'.

Corrective action:

PGCKW completely disables the alarm.

**Empty [Filter] sub-block**

The **[filter]** sub-block has no identifiers.

Corrective action:

PGCKW disables the filter.

**Empty [subcom] sub-block**

The **[subcom]** sub-block has no identifiers.

Corrective action:

PGCKW disables the sub-commutation and treats the view-port as a single sampled parameter.

**Forward & backward plots are ignored if not a lineplot**

The lineplot option is disabled but the **forwardplot** or **backwardplot** options are enabled. The **forwardplot** or **backwardplot** identifiers are only valid when lineplot is on. Note, not specifying the lineplot is equivalent to disabling it.

Illegal:

```
forwardplot = on  
lineplot = off
```

legal:

```
forwardplot = on  
lineplot = on
```

Corrective action:

PGCKW disables the forwardplot and backwardplot option.

**Parameter name defaulted to XXXX**

All three identifiers **mnemonic**, **title** and **units** are all blank or missing, i.e. not specified.

At least one of the three must be present and non-blank for PGCKW to assign a **databaskey** to the view-port.

Corrective action:

PGCKW sets the **mnemonic** to "XXXX"

This does not cause an error on pages 0 and 9999. These can have all three attributes blank as some view-ports on these pages are intentionally invisible, namely the page 0 'tag' parameter.

**Simultaneous Alarm and Parameter recording illegal**

The [valueport] has a parameter recording defined twice - once within a [valueport] block and once within an [alarm] sub-block, i.e. the following line appears twice.

filename = *filename*

There must only be one occurrence of such a statement.

Corrective action:

PGCKW disables BOTH occurrences, i.e. the recording is disabled.

**Super-comm increment is zero with count > 1**

The **supercommincr** identifier specifies a zero byte offset between consecutive super-commutated samples (only legal for single-sampled parameters), but the **supercommcount** is greater than 1 denoting a super-commutated parameter.

Legal:

supercommincr = 0	; only legal value for single sampled parameter
supercommcount = 1	; single sampled

illegal

supercommincr = 0	; only legal value for single sampled parameter
supercommcount = 2	; super-commutated - 2 samples/packet

Corrective action:

PGCKW sets **supercommcount** to 1 denoting a single-sampled parameter.

**x upper limit > x lower - swapping limits**

'x upper limit < x lower - swapping limits'

The **xupper** value is less than the **xlower** value.

Note, if xupper is identical to xlower, PGCKW generates an error, not a warning.

Corrective action:

PGCKW swaps the xupper and xlower values so that  $xupper > xlower$ .

**y upper limit > y lower - swapping limits**

'y upper limit < y lower - swapping limits'

The **yupper** value is less than the **ylower** value.

Note, if yupper is identical to ylower, PGCKW generates an error, not a warning.

Corrective action:

PGCKW swaps the yupper and ylower values so that  $yupper > ylower$ .

## 6.2 Errors

### [bitoperation] specification block is empty

The [bitoperation] sub-block has no mandatory bitmask identifier

A correct specification would look like

```
[calibration]
  function = onoffbit
  [bitoperation]
    bitmask = 0x00000001
```

### Block specifier must be [valueport],[packet] or [system]

The top-level nested block must always be either [valueport], [packet] or [system]. All other block identifiers such as [input] are nested blocks of one of these three top-level blocks. This error commonly occurs when there is unidentified text in the left most column, e.g., a comment was added which was not preceded by a comment character (';' or '#').

Note, [packet], [input] and [system] are reserved for page 0 and not documented in this manual.

### Calibration missing valid function

The mandatory function identifier appearing after the [calibration] block identifier has an invalid function.

#### Example

the following specification has an incorrectly spelt name `polynmial` which is consequently unrecognisable. It should, of course, be `polynomial`.

```
[calibration]
  function = polynmial
  [polynomial]
    a1 = 0.001
    a0 = 0.0
```



**Column not found or outside limits (1 <= col <= 160)**

The `column` identifier is mandatory otherwise TPS does not know where to display the parameter on the page. Its range must be in the confines of the maximum TPS display size.

**Indentation Wrong. Check spacing/tabs (set to 8 ?)**

The page specification language relies heavily on each nested specification block having all statements within a block aligned. Every time a new block is nested within a higher block, it must be indented 1 or more spaces or tabs. It is this indentation that the page compiler uses to determine that the current nested block is a sub-block of a higher level block. Common problems can occur when mixing spaces and tabs for indentation. In particular this problem can be compounded when using different editors with different tab settings.

**Invalid Display Cycle Rate definition**

The `displayrate` identifier has an invalid value. For instance, the following specification is invalid because 'msec' should read 'msecs'.

e.g. `displayrate = "every200msec"`

**Invalid number**

The value `nnnn` specified on the right hand side of the equals sign in a specification line has an one or more invalid characters, i.e. non-numeric.

Example: The value specified for the `byteindex` has an illegal numeric digit 'x'.

```
[valueport]
byteindex = 10x
```

**Missing coefficient, calibration disabled**

The `[polynomial]` sub-block is missing a mandatory polynomial coefficient. As a minimum specification both the `a0` or `a1` coefficients must be defined, coefficients `a2`, `a3` and `a4` are optional. However if, say, coefficient `a3` is given, all lower order coefficients must also be given, i.e., `a0`, `a1` and `a2`.

**Missing lower, upper x,y limit(s), not proceeding**

The `[plotport]` sub-block is missing one or more of the following mandatory identifiers:  
`xlower`, `xupper`, `ylower`, `yupper`.

**No bitmask found for [bitoperation]**

A `[bitoperation]` sub-block is missing a valid **bitmask** identifier.

**No byte index into packet specified**

A `[valueport]` block must always contain a `byteindex` identifier IF there is no `view` identifier given. When `view` is specified, the `byteindex` defaults to 0 if it is not explicitly stated. When `view` is not defined, `view` defaults to the `packetheader` as in `view = packetheader`.

**No datatype or rawformat specified**

A `[valueport]` block must always contain a `datatype` or `rawformat` identifier, without this, TPS cannot determine the field width of the parameter specified by the `view` statement. Note, specifying the `displayformat` is not sufficient since it only specifies the display format and not the raw format of the parameter as embedded in the packet.

**Parametername invalid/missing parameter**

The *parameter* specified in the `view=parameter` statement is unrecognised

**Plotport row outside limits (1 <= row <= 80)**

The plotport `row` identifier must lie within the range 1-80.

**Plotport row not found**

The plotport `row` identifier is a mandatory part of the [plotport] sub-block

**Plotport column outside limits (1 <= column <= 160)**

The plotport `column` identifier must lie within the range 1-160.

**Plotport column not found**

The plotport `column` identifier is a mandatory part of the [plotport] sub-block

**[plotport] specification block is empty**

The [plotport] sub-block has no identifiers specified. The mandatory identifiers required are shown in the below example

```
[plotport]
  row = 21
  column = 2
  ; width = 76      ; optional, default = 38
  ; height = 18     ; optional, default = 11
  xlower = 0.0
  xupper = 10.0
  ylower = 0.0
  yupper = 100.0
```

**[polynomial] specification block is empty**

The [polynomial] sub-block has no mandatory polynomial coefficients `a0` and `a1` defined - coefficients `a2`, `a3` and `a4` are optional.

A correct specification would look like

```
[calibration]
  function = polynomial
  [polynomial]
    a1 = 0.001
    a0 = 0.0
```

**Row not found or outside limits (1 <= row <= 80)**

The `row` identifier is mandatory (as is the `column`). If it is specified, then the row number `nn` specified in `row = nn` must lie in the range 1 to 80.

**Unable to use line. Identifier unrecognised**

This is generated when an identifier is invalid either because it is spelt incorrectly or it is simply not one of the set of valid identifiers allowable in the page file - see section 3.2 for a full list of identifiers.

Unrecognised values on the right handside of an `identifier = value` will also generate this error. e.g.

```
backgroundcolor = purple    ; no such color
datatype = integer          ; missing quotes on "integer"
```

**x upper limit == x lower - not proceeding**

The [plotport] `xupper` limit must not be the same as the `xlower` limit

**y upper limit == y lower - not proceeding**

The [plotport] `yupper` limit must not be the same as the `ylower` limit

## 7 Displayrate and DCWs

A DCW is an unsigned long number, i.e. 32 bits, which control the display rate of a parameter.

Online configuration provides a **Rate** edit item in the view-port settings dialog box which, when the loop-time of TPS is at its default of 10 msec, gives a drop-down list of 6 possible rates.

```
every10msecs
every20msecs
every40msecs
every80msecs
every160msecs
every320msecs
```

These are always internally converted by TPS to Display Control Words, termed DCWs. The rates are converted as follows:

online Rate	offline DCW	
every10msecs	==	F01ST01
every20msecs	==	F02STmm      mm=00 or 01
every40msecs	==	F04STmm      mm =00 to 03
every80msecs	==	F08STmm      mm =00 to 07
every160msecs	==	F16STmm      mm =00 to 15
every320msecs	==	F32STmm      mm =00 to 31

The mm value is a phase randomly computed by PGCKW and TPS for each view-port.

When PGCKW reverse engineers a page, it writes out the displayrate identifier in DCW format and not the original online format.

HOWEVER, you can still offline configure a rate in the "everynmmsecs" format,

```
displayrate = "every10msecs"
```

## DCW details

Each bit represents a TPS loop (10ms by default) thus a DCW represents 32 loops. A loop is the shortest period in which TPS cycles through every function required of it including parameter display and processing. The loop is analogous to the clock cycle of a processor, i.e., one loop is one clock-cycle of TPS. The display of a parameter on a loop-by-loop basis is controlled by the bit settings in the DCW. If DCW = 0x11111111 then, since bit 0 is 1, the parameter is displayed on loop 0. Bits 1-3 are zero, hence, a parameter is not displayed on loops 1-3. With bit 4 also set to one, the parameter is displayed again on loop 4 giving a 40ms pause between the parameter being displayed on loop 0 and 4. With the above bit pattern, the 40ms refresh period for the parameter is maintained. If the bit pattern were 0xffffffff, the parameter would be displayed on every loop, i.e., a 10ms refresh rate. Although any arbitrary bit pattern can be used, it makes little sense to display a parameter asynchronously, e.g., a 10ms refresh rate followed by a 40ms refresh rate etc. Therefore, TPS generally has refresh rates which are sub-multiples of 32 i.e. every 1, 2, 4, 8, 16 or 32 loops (alternatively expressed as every 10, 20, 40, 80, 160 or 320 milliseconds).

The power of the DCW specification lies in the ability to 'phase' two parameters, e.g., display them at the same rate but on different loops. If a parameter is displayed only every four loops, then it can be displayed starting on either the 0, 1st, 2nd or 3rd loop and, thereafter, repeated every four loops. Thus, two parameters with the same loop rate could initially start on different loops.

**Note, phases are internally computed by TPS and cannot be modified by the user.**

Because of the periodicity of refresh rates, there are only a small set of possible loop rates and phases which are listed fully below. In the displayrate = FnnSTmm specification, the nn specifies every nn loops and the mm specifies the starting loop (or 'phase').

The '0xnnnnnnnn' in the 2nd column on the next page is the bit pattern equivalent of the displayrate FnnSTmm identifier option and is shown for information only - it cannot be used as an option.

```
F00ST00 0x00000000 ; no display, often used for text only view-ports
F01ST01 0xffffffff ; every loop
F02ST01 0x55555555 ; every other loop
F02ST02 0aaaaaaaa
F04ST01 0x11111111 ; every fourth loop
F04ST02 0x22222222
F04ST03 0x44444444
F04ST04 0x88888888
F08ST01 0x01010101 ; every eighth loop
F08ST02 0x02020202
F08ST03 0x04040404
F08ST04 0x08080808
F08ST05 0x10101010
F08ST06 0x20202020
F08ST07 0x40404040
F08ST08 0x80808080
F16ST01 0x00010001 ; every 16 loops
F16ST02 0x00020002
F16ST03 0x00040004
F16ST04 0x00080008
F16ST05 0x00100010
F16ST06 0x00200020
F16ST07 0x00400040
F16ST08 0x00800080
F16ST09 0x01000100
F16ST10 0x02000200
F16ST11 0x04000400
F16ST12 0x08000800
F16ST13 0x10001000
F16ST14 0x20002000
F16ST15 0x40004000
F16ST16 0x80008000
F32ST01 0x00000001 ; every 32 loops
F32ST02 0x00000002
F32ST03 0x00000004
F32ST04 0x00000008
F32ST05 0x00000010
F32ST06 0x00000020
F32ST07 0x00000040
F32ST08 0x00000080
F32ST09 0x00000100
F32ST10 0x00000200
F32ST11 0x00000400
F32ST12 0x00000800
F32ST13 0x00001000
F32ST14 0x00002000
F32ST15 0x00004000
F32ST16 0x00008000
F32ST17 0x00010000
F32ST18 0x00020000
F32ST19 0x00040000
F32ST20 0x00080000
F32ST21 0x00100000
F32ST22 0x00200000
F32ST23 0x00400000
F32ST24 0x00800000
F32ST25 0x01000000
F32ST26 0x02000000
F32ST27 0x04000000
F32ST28 0x08000000
F32ST29 0x10000000
F32ST30 0x20000000
F32ST31 0x40000000
F32ST32 0x80000000
```