Formal .pmap File Format Specification

Version: 1.0.0 (Draft)
Date: 2025-09-14
Author: Hans Kramer
Status: Proposal

1. Introduction

1.1 Purpose

The pmap file format is an open-source, text-based, domain-specific data standard for defining the port map of a two-stroke engine cylinder. Its sole purpose is to enable lossless interoperability of 2D port geometry between design, simulation, and manufacturing software.

1.2 Scope

This specification defines the syntax and semantics of the pmap format. It describes a 2D representation of the unwrapped cylindrical surface of a two-stroke engine cylinder. The X-axis represents the circumference (0 → π × nominal bore); the Y-axis represents both the stroke length and the barrel length.

1.3 Design Philosophy

• Open-source: The format should always be available (free of charge) to those whom require it.
• Simplicity: The format uses simple group-code and value pairs.
• Precision: All geometric data is defined with a precision of at least 1.0e-5 units.
• Unambiguity: The specification is strict. Parsers must not guess intent.
• Focus: The format exclusively describes port geometry and essential metadata. It excludes presentational data (colors, text, dimensions).

2. File Structure

```
PMAP_FILE -> HEADER_SECTION ENTITIES_SECTION EOF
```

2.1 Group Codes

Code	Meaning
0	Start of section/entity, or EOF
2	Section name (HEADER, ENTITIES)

3. Header Section

Header variables are defined by group code 9 (variable name) and the following line (value).

3.1 Group Codes

Code	Meaning
9	Header variable name

3.2 Mandatory Header Variables

Variable	Type	Units	Description	Example
$FORMATVERSION	string	n/a	Spec version (semantic versioning). Parser MUST support major version.	1.0.0
$MEASUREMENTUNITS	string	n/a	Valid values: MM, IN (case-insensitive).	MM
$NOMINALBORE	number	declared units	Design bore diameter. Nominal circumference $C = \pi \times$ $NOMINALBORE.	56.0
$NOMINALSTROKE	number	declared units	Design stroke length ($0 \rightarrow$ $NOMINALSTROKE).	50.60
$TOTALCYLINDERLENGTH	number	declared units	Full cylinder length. MUST be $\geq$ $NOMINALSTROKE.	110.0
$EXHAUST_CENTER_X	number	declared units	Exhaust center location along bore circumference. Defines the 0° point on the X-axis.	0.0
$TDC_OFFSET	number	declared units	Distance from deck (Y=0) to piston at TDC. Used to compute BDC position.	0.5

3.3 Optional Header Variables

Variable	Type	Units	Description	Example
$CURRENTBORE	number	declared units	If the design is for an over-bored cylinder, this defines the new target bore diameter.	56.50
$RINGGAP_COUNT	number	n/a	Amount of piston ring gaps defined (0 being top, 1 = second, etc.).	2
$RINGGAP_0_ANGLE	number	degrees	Position of the first piston ring gap in degrees relative to exhaust center location.	45.0
$RINGGAP_0_WIDTH	number	declared units	Width of first ring gap.	0.35
$CREATEDBY	text	n/a	Person whom created this file.	Hans Kramer
$CREATIONDATE		n/a	ISO 8601 timestamp	2025-09-11T14:30:00Z
$DESCRIPTION	text	n/a	Human-readable description of the file.	’99 RS250 Port Map

4. Entities Section

Entities define port geometries. Supported: POLYLINE, ARC, GROUP.

4.1 Entity Name (Group Code 0)

Code	Marker
POLYLINE	Declares the start of a new polyline entity.
ARC	Declares the start of a new arc entity.
GROUP	Declares the start of a new group entity.
ENDPOL	Declares the end of a polyline entity.
ENDARC	Declares the end of an arc entity.
ENDGRP	Declares the end of a group entity.

4.2 Layer Name (Group Code 8)

An optional string to group and identify entities.

Recommended value	Meaning
MAIN_EXHAUST	.
BRIDGED_EXHAUST_LEFT	.
BRIDGED_EXHAUST_RIGHT	.
AUXILIARY_EXHAUST_LEFT	.
AUXILIARY_EXHAUST_RIGHT	.
POWERVALVE	Has flag 18 connected to parent Exhaust
TRANSFER	.
INTAKE	.
BOOST	.
DRAFT	.
FIXTURE	.

4.3 Flags (Group Code 70)

Mandatory integers that define the state and type of the entity using **bit-codes**.
Flags can be added together for complex cases.

- A port open to the bottom that also wraps: 4 (Y-max) + 8 (X-seam) = 12
- The placeholder flag (1) is **mutually exclusive** and should never be combined with closure flags. This means uneven entity flags will never be allowed for future versions.

Value	Marker
0	Closed shape
1	Placeholder
2	Connected to Y-min edge
4	Connected to Y-max edge
8	Connected to seam
18	Open shape connected/ closed by entity
X+X	Combinations allowed except with 1.

4.4 POLYLINE Entity

```
0
POLYLINE
8
<layer_name>
70
<flags>
10
<X1>
20
<Y1>
...
```

4.4.1 Group Codes

Code	Meaning
8	Layer name (optional)
10	X coordinate (vertex)
20	Y coordinate (vertex)

4.4.2 Polyline Flags

Code	Meaning
0	Closed polygon
1	Placeholder
2	Connected to Y-min edge
4	Connected to Y-max edge
8	Connected to seam
18	Open shape connected/ closed by entity
X+X	Combinations allowed except with 1.

4.5 ARC Entity

```
0
ARC
8
<layer_name>
70
<flags>
10
<X_center>
20
<Y_center>
40
<radius>
50
<start_angle>
51
<end_angle>
```

4.5.1 Group Codes

Code	Meaning
8	Layer name (optional)
10	X coordinate (center)
20	Y coordinate (center)
40	Radius (arc)
50	Start angle (arc)
51	End angle (arc)

4.5.2 Arc Flags

Code	Meaning
0	Closed arc (circle)
1	Placeholder
2	Connected to Y-min edge
4	Connected to Y-max edge
8	Connected to seam
18	Open shape connected/ closed by entity
X+X	Combinations allowed except with 1.

4.5.3 Arc Geometry Derivation

The ARC entity defines a circular arc in the Cartesian coordinate system.
The values given by group codes 10 and 20 specify the center point of the arc, not a vertex of the curve.
The radius (40) defines the distance from the center to all points on the arc.
The start (50) and end (51) angles are expressed in degrees, measured counter-clockwise from the positive X-axis, consistent with conventional CAD polar coordinates.
The start point and end point of the arc are computed as

```
X_start = X_center + Radius × cos(StartAngle)
Y_start = Y_center + Radius × sin(StartAngle)

X_end   = X_center + Radius × cos(EndAngle)
Y_end   = Y_center + Radius × sin(EndAngle)
```

Validation rules that depend on entity closure (see sec. 7) must use these computed points.
Example:

```
0
ARC
10
8.0
20
46.2
40
6.0
50
180.0
51
270.0
0
ENDARC
```

Represents an arc from (X 2.0, Y 46.2) to (X 8.0, Y 52.2), centered at (8.0, 46.2) with R= 6.0.

4.6 GROUP Entity and Nesting

```
0
GROUP // Start of a parent group
0
POLYLINE
... // Vertex data for polyline
0
ENDPOL
   0
   GROUP // Start of a nested group
   0
   ARC
   ... // Arc data
   0
   ENDARC
0
ENDGRP // Ends the nested group
0
ENDGRP // Ends the parent group
```

The GROUP entity is a container that allows for the logical organization of one or more entities (POLYLINE, ARC, or nested GROUP). This is useful for defining complex port shapes composed of multiple primitives (e.g., a transfer port made of a set of polylines and arcs) and for applying common properties (Layer, Flags) to a set of entities through inheritance.

4.6.1 Group Composition

A Group (0 GROUP ... 0 ENDGRP) may contain:
One or more POLYLINE entities.
One or more ARC entities.
One or more nested GROUP entities.
Groups can be recursively nested to an arbitrary depth.

4.6.2 Closure Inside Groups

Each entity inside a group does not obey its own closure rules.
Entities must use their explicit end markers:
0 ENDPOL for polylines.
0 ENDARC for arcs.
Repetition of the first vertex alone is not considered valid closure.

4.6.3 Nesting Semantics

Groups may contain other groups, which may themselves contain groups, without limit.
A parser must handle recursive descent until the matching ENDGRP is found.
Improperly terminated or overlapping group markers (GROUP without a matching ENDGRP) must cause the file to be rejected.

4.6.4 Layer & Flag Inheritance

Groups may declare a Layer (group code 8) and Flags (group code 70).
Entities inside a group may not declare their own Layer and Flags.
Inheritance rules:
If an entity inside a group inherits the Layer of its parent Group.
If an entity inside a group inherits the Flags of its parent Group.
This inheritance also applies recursively through nested groups.

4.6.5 Validation Rules for Groups

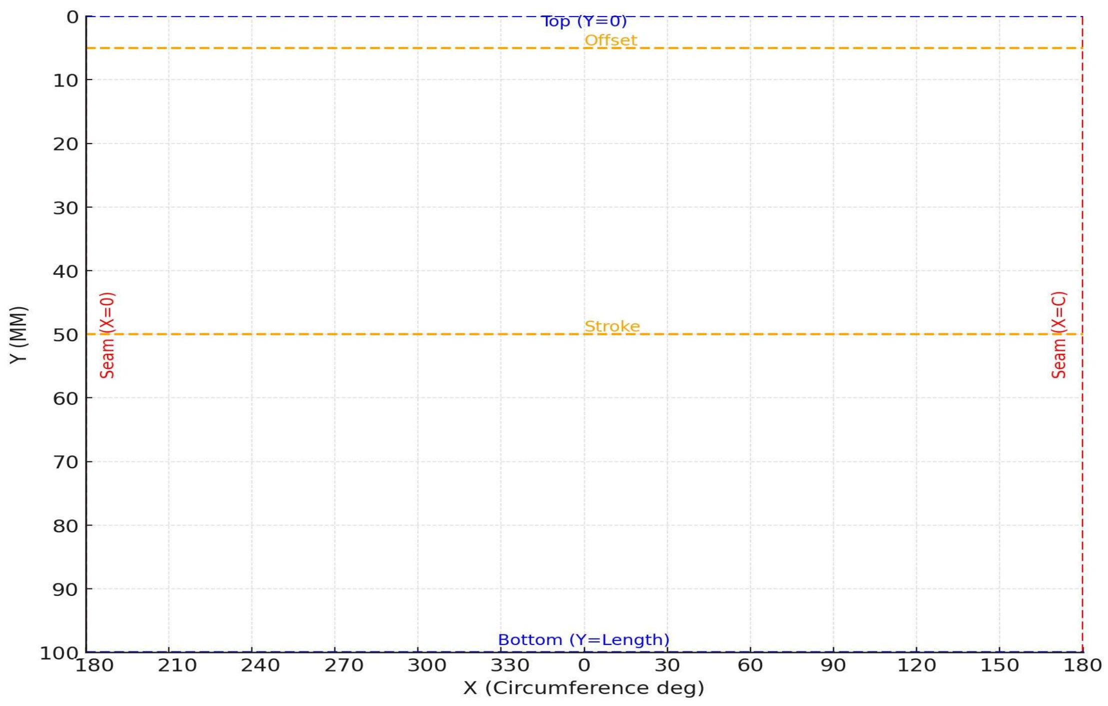
A Group is considered valid if:
All contained entities are valid.
All nested groups are properly closed.

5. Coordinate System

All coordinates and dimensions within the pmap format are defined in the linear units declared in the $MEASUREMENTUNITS header variable (MM or IN).
This choice is fundamental to ensuring consistent precision and interoperability across different cylinder designs. Using linear units (e.g., millimeters) for both the circumferential (X) and axial (Y) axes, rather than expressing the X-axis in angular degrees, provides a critical advantage: it guarantees uniform resolution regardless of the cylinder's bore diameter.

For example, on a large-bore cylinder, a single degree of circumference represents a much larger linear distance than on a small-bore cylinder. Defining ports using angular coordinates would therefore result in a loss of precision and manufacturing consistency for larger engines. By using linear units, a vertex positioned at X=1.5 always represents the same physical distance from X=1.4 on any cylinder, ensuring port geometry is defined with consistent, high fidelity.

X-Axis: Represents the circumference of the cylinder. The valid range is from 0.0 to C, where C = π * $NOMINALBORE and the coordinate 0.0 defines the intersection of the top of the cylinder barrel and the $EXHAUST_CENTER_X.
Y-Axis: Represents the cylinder length along the cylinder's axis. The value 0.0 typically represents the top of the barrel (cylinder deck). The value $TOTALCYLINDERLENGTH represents the bottom of the cylinder. The points (0.0, Y) and (C, Y) are adjacent, representing the same physical seam on the cylinder.
TDC offset defines piston-at-TDC reference where negative values represent piston protrusion relative to the top of the cylinder. BDC = $TDC_OFFSET + $NOMINALSTROKE.



6. Comments

A comment is text that should be ignored by parsers, but is useful for programmers. Comments are normally used to annotate code for future reference. A parser treats them as white space.

A comment is written in one of the following ways:

- The /* (slash, asterisk) characters, followed by any sequence of characters (including new lines), followed by the */ characters.
- The // (two slashes) characters, followed by any sequence of characters. A new line not immediately preceded by a backslash terminates this form of comment. Therefore, this is called a "single-line comment."

The comment characters (/*, */, and //) have no special meaning.

```
0                     // Start of a section marker
SECTION               // Section begins
2                     // Start of a section marker
HEADER                // Header section begins
9                     // Group code for variable
$FORMATVERSION        // Variable name
1.0.0                 // Variable value (string) → format version
/* A comment
with multiple
lines are
ignored when
properly opened
and closed */
9
$UNITS
MM                    // Units (MM or IN)
```

7. Validation Rules

A conforming parser MUST implement the following validation checks:
• File Structure: The file must contain exactly one HEADER and one ENTITIES section, in that order, terminated by EOF.
• Mandatory Header Vars: The parser must reject the file if any mandatory header variable ($FORMATVERSION, $MEASUREMENTUNITS, $NOMINALBORE, $NOMINALSTROKE, $TOTALCYLINDERLENGTH, $EXHAUST_CENTER_X, $TDC_OFFSET) is missing or has an invalid value.
• Entity Closure: For an entity to be valid (i.e., not a placeholder), it must be "closed". The parser must check this based on its flags:

- Flag 0: The first and last vertex (or derivatives for arcs) must be identical (within a tolerance of 1.0e-5 units).
- Flag 2: The first or last vertex (or derivatives for arcs) must have a Y-coordinate <= 0.0 (within tolerance).
- Flag 4: The first or last vertex (or derivatives for arcs) must have a Y-coordinate >= $TOTALCYLINDERLENGTH (within tolerance).
- Flag 8: The entity is defined as closed via the cylindrical seam. No specific vertex check is required, but the entity should meaningfully use the seam (e.g., have vertices near X=0 and X=C).
- Flag 18: The entity is defined as closed if start and end point are connected to a valid closed parent entity.

.
• Non-Overlapping Geometry: Closed entities (i.e., entities with flags 0, 2, 4, or 8) must not overlap in the 2D plane. Overlapping geometry represents an ambiguous or impossible physical state in the manufactured cylinder and must be rejected by the parser.
- Exception: This rule does not apply to entities with flag 18 (Open shape connected/closed by entity), as their purpose is to connect to and be closed by a parent entity, which may involve designed overlaps or intersections (typically used for powervalves).
- Validation: The parser must check that the interior regions of any two closed entities (POLYLINE, ARC, or GROUP) do not intersect. A tolerance of 1.0e-5 units should be used for point-in-polygon and intersection calculations.
- Rationale: Overlapping closed shapes would create ambiguous instructions for machining (e.g., which contour defines the final port edge). This rule ensures the model defines a clear, manufacturable result.
• Placeholder Handling: Entities with flag 1 must be parsed but flagged to the user as incomplete. They should be ignored for simulation or manufacturing purposes
• Use tolerance of 1.0e-5 units for comparisons.

8. Examples

8.1 Minimal Valid File

```
0
SECTION
2
HEADER
9
$FORMATVERSION
1.0.0
9
$MEASUREMENTUNITS
MM
9
$NOMINALBORE
56.0
9
$NOMINALSTROKE
50.6
9
$TOTALCYLINDERLENGTH
110.0
9
$EXHAUST_CENTER_X
0.0
9
$TDC_OFFSET
0.5
0
ENDSEC
0
SECTION
2
ENTITIES
0
EOF
```

8.2 Arc Example

```
0
SECTION
2
ENTITIES
0
ARC
8
BOOST
70
0
10
28.2
20
20.0
40
5.0
50
0.0
51
90.0
0
ENDSEC
0
EOF
```

Appendix A: Versioning Policy

Major (X.0.0): Breaking changes
Minor (1.X.0): Backwards-compatible additions
Patch (1.0.X): Bug fixes / clarifications