PL File Structure

Proprietary and Confidential

Version 3.0

Revised and Updated: January 28, 2014

Pointer Telocation Ltd. 14 Hamelacha St., Rosh Ha'ayin 48091, Israel • Tel: 972-3-5723111 • Fax: 972-3-5723100 • www.pointer.com

Table of Contents

1 Introduction

This document contains a description of the PL file structure, in order to enable a SW programmer to understand, integrate and use PL files in their SW environment.

1.1 Revision History

Version	Date	Author	Description
1.0.	1/1/07	Igor Rogov	Initial draft
2.0	30/3/09	Nessy Turgeman	Second edition
2.1	27/05/09	Nessy Turgeman	Editing
3.0	10/1/14	Nessy Turgeman	Editing

2 File Structure

The PL file is a hex file which contains two major sections:

- EEPROM map (in variant size).
- PL Parameter text, which includes the textual description of all PL parameters.

2.1 EEPROM Section

In order to read the EEPROM map from the PL file, follow the instructions according to the byte structure in the following table:

Byte	Name	Description	Size
1-4	Header	Protected condition [2 bytes]: check if there is a section of protected bytes in the PL (0 – no, 2 - yes)	2
		Size of protected data [2 bytes]: the size of the following protected bytes (in 8 byte resolution) (0 – no bytes)	2
		Protected data – depends on size (8 byte sections)	0-8
5-8	Program start	EEPROM MAP low address	4
9-12	Program end	EEPROM MAP high address	4
13	Include map	Include EEPROM Map flag – Boolean value; if true then read the EEPROM map from start address until the end address	1
14- (size-1)	EEPROM map	Calculate map size by (Program end – Program start)	Map size
14 + size	Zero block	16 bytes of zeros, to distinguish between EEPROM map and parameters	16
	Parameters		

2.2 Parameters Section

Each parameter contains a set of fields; some of the fields are limited by size of bytes and some are limited by the value "0x00" in hexadecimal at the end of the field.

Index	Limitation	Name	Description	Size
1	'00'	Name	Text name of the parameter	Unknown
2	Size	Parameter type	See the *Parameter Types* table	1

Index	Limitation	Name	Description	Size
3	'00'	Help	Help information regarding the parameter	Unknown
4	'00'	Comment		Unknown
5	Size	Protect	Boolean value; true if the parameter is protected by a password	1
6	'00'	Password	Password for protected parameter	Unknown
7	'00'	Units	Measurement units text	Unknown
8	Size	Address	Parameter address in the EEPROM map	4
9*	Size	Size	Parameter size in the EEPROM map	4
10	Size	Factor	Parameter factor constant to be displayed	8
11	Size	Created	Date Time format	8
12	Size	Modified	Date Time format	8
13	Size	Accessed	Date Time format	8
14	Size	Enumeration counter	Represents the number of enumerations in the specified parameter	4
15**	'00'	Enumerations	Each enumeration will be followed by a '00'	Unknown
16***	Size	Parameters counter	Represents the number of parameters in the PL file; the last parameter must have the value of 0	4

- *** Index [9]**

  For Bit script parameter (type 18) the size represent start bit (Size / 10) and bits length (Size module 10).

  Example: Size 17 means start bit 1 and bits length 7.

  For Long bit script parameter (type 21) the size represents the start bit (3 LSB bits) and bits length (5 MSB bits of the LSB byte + 1, min bits length is 1).

  Example: Size 52 (binary 00110100) means start bit 4 (binary 100) and bits length 7 (binary 00110 + 1).

- **** Index [15]**

  Enumeration gets the value according to its index; the enumeration value can be overwritten if the following format is used in the enumeration string: #X#S

This means that the value X will be assigned to the enumeration and the enumeration string will be S.

A list of null values terminates the string (list length is defined in the Enumeration counter).

- ***** Index [16]**

  Represents the number of "child" parameters.

2.3 Parameter Procedure

The following is a set of sequential steps to read all parameters:

1 Name (read until the sign #0)

2 Parameter Type (1 byte)

3 Help (read until the sign #0)

4 Comment (read until the sign #0)

5 Protect (size 1 byte) – Boolean value

6 Password (read until the sign #0)

7 Units (read until the sign #0)

8 Address (size 4 bytes)

9 Size (size 4 bytes)

10 Factor Const (size 8 bytes)

11 Created (size of TDateTime = 8 bytes)

12 Modified (size of TDateTime = 8 bytes)

13 Accessed (size of TDateTime = 8 bytes)

14 Enumeration Counter (size 4 bytes) – the counter for Enumeration; when counter>0 use the following:

Enumeration[I]:=(read from stream until the sign #0);

15 Child Counter (size 4 bytes) - the counter of children repeats (read parameters from stream – 15 steps) until counter >0.

2.4 Parameter Types

The following table lists all parameter types and their values:

Name	Value (decimal)
Root	0
Directory	1
Undefined	2

Name	Value (decimal)
Flag	3
Decimal	4
Binary	5
Hexadecimal	6
Coordinate	7
GSM Service Center Address	8
GSM Destination address	9
IP	10
PIN	11
Null Terminated string	12
Dallas	13
Anti String	14
ASCII	15
Unix IP	16
*Way Points	17
Bit script	18
Edge way point	19
**Signed Decimal	20
Long Bit script	21
Date time	22

- * Way point and Edge way point structures are defined in the *Programming Manual*.
- ** 2-complement decimal value

2.5 Extraction of Hardware and Firmware Versions from the PL File

HW and FW version values are included in "Root" (parameter type = 0):

- HW version = Size [4 bytes]
- SW version = Factor [8 bytes] and address [4 bytes]
  - Software low version = Factor div 100
  - Software high version = Factor mod 100
  - Software low sub version = Address div 100

- Software high sub version = Address mod 100

2.5.1 Firmware Version Example

FW version XX.YY → xx = Factor YY = Address

1. Address: 67 00 00 00
   - Invert it to Intel wise: 00 00 00 67
   - Convert hex to dec 67 → 103
   - High FW sub = 103 modulu (100) = 3
   - Low FW sub = 103/100 = 1
2. Factor: 00 00 00 00 00 4E A5 40
   - Invert it to Intel wise: 40 A5 4E 00 00 00 00 00
   - Convert it to double.(for example 203.658)
   - Round it (203)
   - High FW = 203 modulu (100) = 3
   - Low FW = 203/100 = 2

The final values:

- From FW low = 2.1
- To FW high = 3.3

2.5.2 Code Example in Delphi

```
type
TParameterTypes = (ptRoot, ptDirectory, ptUndefi ned, ptFlag, ptDec, ptBin,
   ptHex, ptCoordinate, ptSCA, ptDA, ptIP, ptPIN, ptNTString, ptDallas,
   ptAnsiString, ptASCII,
   ptUni xIP, ptWayPoint, ptBitscript);
end;


procedure TMainFor m.LoadParamFile;
var
   Stream: TStream;
   TempW: Word;
   I: Integer;
   IncludeEepromMap: Boolean;


procedure Error;
begin
   raise Exception.Create( LoadResStri ng(@SInvalidParamLib));
```

```
end;


procedure LoadParamData;

begin

   if Stream.Read(ProgramLowAddress, SizeOf( Integer)) <> SizeOf(Integer)
then Error;

   if Stream.Read(ProgramHighAddress, SizeOf(Integer)) <> SizeOf(Integer)
then Error;

   if (UnitType < 1) and (ProgramHi ghAddress > 255) then begin

   ProgramHi ghAddress:= 255;

   MessageDl g( LoadResStri ng(@SMemor yRangeTooLarge), mtInfor mation,
[mbOK], 1);

end;

   if Stream.Read(Incl udeEepromMap, SizeOf(Boolean)) <> Si zeOf(Boolean)
then Error;

   if IncludeEepromMap then

   if Stream.Read(InternalEepromMap[0], SizeOf(TEepromMap)) <>
   SizeOf(TEepromMap)

then Error;

   FItemCash.LoadFromStream(Stream); end;


begin

IncludeEepromMap:= Incl udeEepromContent.Checked; Stream:=
TFileStream.Create(ParamFileName, fmOpenRead);

try

   if Stream.Read(TempW, Si zeOf(Word)) <> SizeOf( Word) then Error;

   case TempW of

1: begin

   LoadParamData;

         RefreshExecute(nil); end;

2: begin

   if Stream.Read(TempW, Si zeOf(Word)) <> SizeOf( Word) then Error;
   SetLength(ProtectedBlocks, TempW) ;

   for I:= 0 to TempW-1 do

   begin

   with ProtectedBlocks[I] do

   begin
```

```
   if Stream.Read(FromAddr, Si zeOf(Cardinal)) <> SizeOf(Cardinal) then
Error;
   if Stream.Read(ToAddr, SizeOf(Cardi nal)) <> SizeOf(Cardi nal) then Error;
   end;
   end;
   LoadParamData; RefreshExecute(nil); end;
   else MessageDlg( LoadResString(@SInvalidParamLibVersion), mtError,
[mbOK], 1);
end;
finally Stream.Free; end;
   end;

procedure TParameterItem.LoadFromStream;

   function ReadNullStr: string;
   var
   Ch: Char; begin Result:= ''; repeat
   Stream.Read(Ch, 1); Result:= Result + Ch; until Ch = #0;
   Delete(Result, Length(Result), 1);
end;

var
   Counter: Integer;
Begin
   ClearChildren;
   FName:= ReadNullStr;
   Stream.Read(FParameterType, 1);
   FHelp:= ReadNullStr;
   FComment:= ReadNullStr;
   Stream.Read(FProtect, 1);
   FPassword:= ReadNullStr;
   FUnits:= ReadNullStr;

   Stream.Read(FAddress, SizeOf(Integer));
   Stream.Read(FSize, Si zeOf( Integer));
   Stream.Read(FFactorConst, Si zeOf(Double));
```

```
   Stream.Read(FCreated, SizeOf(TDateTi me));
   Stream.Read(FModi fied, SizeOf(TDateTi me));
   Stream.Read(FAccessed, SizeOf(TDateTi me));
   if FParameterType=ptBitscript then
   begin
   FStartBit:= FSi ze div 10;
   FBitSize:= FSi ze mod 10;
   FSize:= 1;
   end
   else
   begin
         FStartBit:=0;
   FBitSize:=8; end;

   Stream.Read(Counter, SizeOf( Integer));
   while Counter > 0 do
   begin
   FEnumeration.Add(ReadNullStr);
   Dec(Counter);
end; FState:= [];
Stream.Read(Counter, SizeOf( Integer));
while Counter > 0 do
begin
   AddChild('', ptUnDefined).LoadFromStream(Stream); Dec(Counter);
end;
FChangeMade:= False;
end;

procedure TMainFor m.SaveParamFile;
var
   Stream: TStream; IncludeEepromMap: Boolean; TempW: Word;
begin
   IncludeEepromMap := Incl udeEepromContent.Checked;
   Stream:= TFileStream.Create(ParamFileName, fmCreate); try
   TempW:= 2;
```

```
   Stream.Write(TempW, Si zeOf(Word)); TempW:= Length(ProtectedBlocks);
   Stream.Write(TempW, Si zeOf(Word));

   TempW:= 0;

   while TempW < Length(ProtectedBlocks) do with ProtectedBlocks[TempW] do

   begin

   Stream.Write(FromAddr, SizeOf(Cardinal)); Stream.Write(ToAddr,
   SizeOf(Cardinal)); Inc(TempW);

   end;
Stream.Write(ProgramLowAddress, SizeOf( Integer));

Stream.Write(ProgramHi ghAddress, SizeOf(Integer));

Stream.Write(Incl udeEepromMap, SizeOf(Boolean));

if IncludeEepromMap then Stream.Write(InternalEepromMap[0],
SizeOf(TEepromMap)); FItemCash.SaveToStream(Stream);

finally

Stream.Free;

end;

end;

procedure TParameterItem.SaveToStream;

var

   Counter: Integer;

Buffer: string;

begin

   Buffer:= FName + #0 + Char(FParameterType) + FHelp + #0 + FComment + #0 +
   Char(FProtect) + FPassword + #0 + FUnits + #0;

Stream.Write(Buffer[1], Length(Buffer));


Stream.Write(FAddress, SizeOf(Integer));

if FParameterType=ptBitscript then begin

   FSize:=FStartBit*10+FBi tSize;

end;

Stream.Write(FSize, SizeOf(Integer)); Stream.Write(FFactorConst,
SizeOf(Double)); Stream.Write(FCreated, SizeOf(TDateTi me));
Stream.Write(FModified, SizeOf(TDateTi me)); Stream.Write(FAccessed,
SizeOf(TDateTi me));


Counter:= FEnumeration.Count; Stream.Write(Counter, SizeOf(Integer));

for Counter:= 0 to FEnumeration.Count-1 do begin
```

```
   Buffer:= FEnumeration[Counter] + #0; Stream.Write(Buffer[1],
Length(Buffer)); end;

Counter:= FChildren.Count; Stream.Write(Counter, SizeOf(Integer)); for
Counter:= 0 to FChildren.Count-1 do

   TParameterItem(FChildren.Items[Counter]).SaveToStream(Stream);

end;
```