LaCoGen – V1.2 File Format (.lacobj files)

Document author: Duncan Munro
duncan@duncanamps.com

Document date: Thursday, 5 May 2022

Document version: 1.2

Contents

1 Introduction

The .lacobj file is a compiled binary file created by LaCoGen from a .lac input file. While the contents of the .lac file are human readable, the contents of the .lacobj file are not.

1.1 Intended Use

It is intended that the .lacobj file can be read from disk into a TLCGParser instance or can be stored as part of the program's resources.

1.2 Data Conventions

The following data types are used in the format:

Data Type	Size in bits	Detail
Uint8	8	Unsigned 8-bit integer
Uint16	16	Unsigned 16-bit integer
Uint32	32	Unsigned 32-bit integer
String	Variable	Prefixed by a 16-bit little endian size value, the string is followed by a sequence of bytes representing the string and finished with a NUL terminator. Note that the size is bytes stored and not the string length. The string is stored as a sequence of UTF-8 characters

An example of a string would be to store the value €5 would be:

```
$04 $00 $E2 $82 $AC $35 $00
|       |               | |
|       |               | +-- Terminator
|       |               +----- Digit "5"
|       +--------------- UTF-8 sequence for €
+-------------------- Size of string, not include size itself or terminator
```

A $ symbol is used to denote hexadecimal values in the text.

1.3 High Level Structure

The high-level structure follows the form:

- Block
  - Optional Record
  - Optional Record
  - Optional Record
- Block
  - Optional Record
  - Optional Record
  - : :

The Block part contains nothing more than a magic word and the block type number.

1.4 Block Reference

The major blocks are:

- Document Details
- Dictionary
- Tokens
- DFA Table
- Reduction Rules
- LALR Table

2 Document Details

The header consists of the following components:

Item	Data	Description
Magic word	Uint32	Contains the value $0143414C (the text “LAC” followed by 1 to signify the version number of the software which produce the file
Block ID	Uint8	$01
Record count	Uint32	Number of records following this block

The records following the header provide descriptive information about the compiled form. The following record types may be used and presented in any order:

Item	Data	Description
Record ID	Uint8	$11
Parameter Name	String	The item name of the parameter, for example, AUTHOR or VERSION
Parameter Value	String	The parameter item value in string format. For a parameter expecting a Boolean value, the words TRUE or FALSE in upper case will be used

For some parameters it may be possible to specify them multiple times. The full table of allowed parameter names is as follows

Parameter Name	Type	Mult	Default	Description
%AUTHOR	String	No		Author of the work
%COMMENTBLOCK	String	Yes		Comment block text. This is formatted as delimiter, starting text, delimiter, ending text. Any delimiter can be used if it does not form part of the starting or ending text. For example, C style comments could be specified as |/*|*/ and PASCAL could be specified as two lines; >{>} followed by .(*.*) The maximum length of a start or end comment text is 16 characters, so the total of this entry cannot exceed 34 characters
%COMMENTLINE	String	Yes		Text which starts a comment in a line being processed. For example, or //
%COMMENTMATCH	String	No	TRUE	Specifies if multiple block comments must match, for example if (* must match a closing *) as opposed to a closing }
%COMMENTNESTED	String	No	FALSE	Specifies if block comments can be nested
%COMMENTQUOTE	String	Yes		Specifies if comments should be ignored in quotations. The format can be empty to

				allow comments in quotes to terminate a comment Block. If present, it can take the format of optional escape prefix followed by the character itself. For Pascal there would be two lines '' and "" as the quote characters are escaped by themselves. For C it would be \"
%COPYRIGHT	String	No		Copyright message
%LICENSE	String	No		Any licensing details
%START	String	No		Starting non-terminal, included for info; not needed for the parser to work
%TITLE	String	No		Title of the work
%VERSION	String	No		Version of the document

3 Dictionary

The dictionary header consists of the following components:

Item	Data	Description
Magic word	Uint32	Contains the value $0143414C (the text “LAC” followed by 1 to signify the version number of the software which produce the file
Block ID	Uint8	$02
Index Size	Uint8	Size of Character fields in sections 3.1 and 3.2. Can be 1, 2, 3, or 4, although 4 is unlikely as 3 will cater for 16 million Unicode characters
DICTRECORDS	UintX	Number of records following this block

The records following the header provide descriptive information about the compiled form. The following record types may be used and are presented in a specific order that will also be followed by the DFA. Typically, the record sequence would be characters first in sorted order of lowest to highest, followed by the set ranges ordered by the start character

3.1 Dictionary Character Record

Item	Data	Description
Record ID	Uint8	$21
Character	UintX	The character which forms part of the dictionary as a Unicode representation. For example, the character A would be $00000041, and the Euro symbol would be $000020AC. The number of bytes used to represent this value is defined in section 3 under Index Size

3.2 Dictionary Range Record

The dictionary range record is no longer produced from Version 1.2 of the software as internal processing no longer facilitates this.

Item	Data	Description
Record ID	Uint8	$22
Character From	UintX	The character which forms the start of the range as Unicode. The number of bytes used to represent this value is defined in section 3 under Index Size
Character To	UintX	The character which represents the end of the range as Unicode. The number of bytes used to represent this value is defined in section 3 under Index Size

4 Token block

The token header block consists of the following components:

Item	Data	Description
Magic word	Uint32	Contains the value $0143414C (the text “LAC” followed by 1 to signify the version number of the software which produce the file
Block ID	Uint8	$03
TOKENS	Uint32	Number of records following this block

The records following the header provide descriptive information about the compiled form.

4.1 Token Record

Item	Data	Description
Record ID	Uint8	$31
Ignore flag	Uint8	Value $00 if the token is to be processed or non-zero if the token should be ignored during processing (for example a comment)
Token Name	String	The name of the token in a text form

5 DFA Transition Block

The DFA Transition header consists of the following components:

Item	Data	Description
Magic word	Uint32	Contains the value $0143414C (the text “LAC” followed by 1 to signify the version number of the software which produce the file
Block ID	Uint8	$04
Index Size	Uint8	Size of Transition Array elements in section 5.1. Can be 1, 2, 3, or 4
DFARECORDS	Uint32	Number of records following this block

The records following the header provide descriptive information about the compiled form.

5.1 DFA Transition Record

Item	Data	Description
Record ID	Uint8	$41
Accept Token	Uint32	The accept token from 0 to n-1 where n is the number of different tokens which can be returned). If an entry is not used the value is $7FFFFFFF
Transition Array	UintX x n	An array of transition records corresponding to the number of dictionary entries (DICTRECORDS). If an entry is not used the value is $7FFFFFFF, $FFFFFF, $FFFF or $FF depending on byte size used

6 Reduction Rule Block

The Reduction Rule header consists of the following components:

Item	Data	Description
Magic word	Uint32	Contains the value $0143414C (the text “LAC” followed by 1 to signify the version number of the software which produce the file
Block ID	Uint8	$05
RULES	Uint32	Number of records following this block

The records following the header provide descriptive information about the compiled form.

6.1 Reduction Rule Record

Item	Data	Description
Record ID	Uint8	$51
Head Token	Uint32	Token ID of the head, used when performing a reduction
Rule Count	Uint32	Number of rule elements following the head, used to pop elements off when performing a reduction. Can be zero for the epsilon type rule
Rule ID	String	A tokenised form of the rule, for example: EXPR_MULO_STAR_NUMB
Rule Text	String	The rule text, for example, <Expression> : <MulOp> * Number
Procedure Name	String	This could be specified as part of the .lac file, for example, procaddnumbers or automatically generated by LaCoGen as the Rule ID prefixed by PROC_, for example, PROC_EXPR_MULO_STAR_NUMB

7 LALR Table Block

The LALR Table header consists of the following components:

Item	Data	Description
Magic word	Uint32	Contains the value $0143414C (the text “LAC” followed by 1 to signify the version number of the software which produce the file
Block ID	Uint8	$06
Outcome Size	Uint8	Size of Outcome field in 7.1 in bytes. Can be 1, 2, 3, or 4
LALRENTRIES	Uint32	Number of records following this block

The records following the header provide descriptive information about the compiled form.

7.1 LALR Table Record

An array with LALRENTRIES rows and TOKENS columns which specifies the state transitions.

Item		Data	Description
Record ID		Uint8	$61
Repeat x TOKENS	Entry Type	Uint8	Entry type which can be one of the following • $00 Undefined • $01 Error • $02 Shift to the next outcome • $03 Goto the next outcome • $04 Reduce using rule indexed by the outcome • $05 Accept condition
	Outcome	UintX	Outcome which can be a rule reduction index (Reduce)or a new state to go to (Shift / Goto). Can be Uint8, 16, 24 or 32 as defined in Outcome Size listed in section 7

Appendices

Block ID codes

Block ID	Block description
$01	Header block
$02	Dictionary block
$03	Token block
$04	DFA Transition block
$05	Reduction Rule block
$06	LALR Table block

Record ID codes

Record ID	Record description
$11	Parameter
$21	Dictionary character definition
~~$22~~	~~Dictionary character range definition~~ No longer used
$31	Token definition
$41	DFA table row
$51	Reduction rule record
$61	LALR table record