RN005
Release note

ST Assembler-Linker
ASM 4.52, LYN 3.19, OBSEND 2.15

Introduction

About this release note

This release note contains additional information about the tools, and information on known errors in the tools and on known errors or omissions in the documentation provided with this version.

Release notes are updated periodically in order to keep you abreast of new features and any limitations found in this release. Check the ST web site at *www.st.com* to ensure that this is the latest version of this release note.

Changes in this version of the release note

New features and corrections	The size of OBSEND input and output file full path names has been extended to 260 characters.

Customer support

For more information or help concerning STVD, please contact the nearest sales office. For a complete list of ST offices and distributors, please refer to *www.st.com*.

Software updates

You can download software updates and all the latest documentation from the ST microcontroller support site at *www.st.com.*

Contents

1 Read me first

Host PC system requirements

PC or compatibles running with Windows 2000, Windows XP, and Vista® operating systems.

Note: 1 To install the STVD toolset, which includes the ST Assembler-Linker, and to connect emulators to the USB port for the first time, you must have administrator privileges. Power user or administrator privileges are required to run STVD.

2 Under the Vista operating system, you must run STVD as an administrator. You can either turn off the User Account Control feature (not recommended), or use the ***Run as Administrator*** *command from the contextual menu, then* ***Allow*** *execution.*

2 What’s new in this release?

2.1 Release summary

This release of the ST Assembler-Linker includes the following components.

Updated from previous release:

- The OBSEND object formatter program release 2.14 (obsend.exe) is replaced by OBSEND program release 2.15 (obsend.exe)

Unchanged from previous release:

- The ASM assembler program release 4.52 (asm.exe)
- The LYN linker program release 3.19 (lyn.exe)
- The ABSLIST list file post processor release 1.01 (abslist.exe)
- The LIB librarian program release 2.02 (lib.exe)
- The MSCI post processor 1.05 (mscist7.exe)

2.2 New features and corrections

OBSEND release 2.15

The size of OBSEND input and output file full path names has been extended to 260 characters.

3 Known problems/limitations

3.1 ASM limitations

3.1.1 Limitations for STM8 assembler

- The mnemonics of the following STM8 instructions:
  ```
  ADD SP,#byte
  SUB SP,#byte
  ```
  have been replaced by:
  ```
  ADDW SP,#byte
  SUBW SP,#byte
  ```
- `.h` and `.l` are not supported; therefore, instead of using:
  ```
  ld a,#label1.h
  ld a,#label1.l
  ```
  use:
  ```
  ld a,#{high label1}
  ld a,#{low label1}
  ```
- There is also no suffix to access the most significant byte (extension byte). To get the extension byte, use:
  ```
  ld a,#{low {seg label1}}
  ```
  For example, if label1 address is $112233
  ```
  ld a,#{low {seg label1}}
  ld a,#{high label1}
  ld a,#{low label1}
  ```
  is equivalent to:
  ```
  ld a,#$11
  ld a,#$22
  ld a,#$33
  ```

3.1.2 Limitations for all assemblers

- The assembler may not be able to compute correctly the addresses of labels set by EQU and CEQU to expressions that include more than one label.

A relative label is a label that has been defined in a relative segment that is a segment declared without specifying the start address of the segment, for example `func` is a relative label:

```
segment 'rom'
func
          nop
```

Example:

```
          segment byte at 0080 'ram0'
var0:     ds.b   1
var1:     ds.b   1
          segment 'ram1'
var2:     ds.b   1
var3:     ds.b   1
equ1      equ      {var0+var1}
equ2      equ      {var1+var2}
equ3      equ      {var2+var3}
equ4      equ      {-var2}
equ5      equ      {{2 mult var2}+1}
```

Only the addresses of `equ1` and *`equ2`* can be computed correctly.

More generally, the computation of an expression may fail if it involves labels defined in relative segments, unless it respects the following syntax: it can involve only one relative label in an expression of the form <relative_label> +/- <absolute_quantity>.

A difference of two labels in the same segment is absolute, therefore an expression can be made of a difference of two relative labels provided they have been defined in the same segment.

If the two labels are not in the same segment, the difference may not be computed correctly as in the example below:

```
segment 'ram0'
var1:     ds.b    1
var2:     ds.b    1
segment 'ram1'
var3:     ds.b    1
var4:     ds.b    1
segment 'rom'
          ld  a,{var3-var1}
          ld  a,{var4-var2}
```

`ld a,{var4-var2}` will be erroneously assembled as B600 without taking into account the start addresses of segments '`ram0`' and '`ram1`'.

- Instructions with operands involving a computation on a label in page 0, resulting in an address outside page 0, may be incorrectly assembled.

  See the following case: A variable `var1`, located in zero page, is defined in *file1* but used in *file2*. The operand resulting from an operation on `var1` (an increment in this case) does not fit in zero page.

  A short addressing mode is incorrectly used in the executable instead of a long addressing mode, however the list file with absolute addresses is correct.

  file1.asm:

  ```
      BYTES
      segment byte at ff 'ram0'
  .var1 DS.b 1
  ```

  file2.asm:

  ```
      EXTERN  var1.b
      WORDS
      segment byte at E000 'rom'
      LD {var1+1},A
  ```

  `LD {var1+1},A` is assembled

  – in list file with absolute addresses as: `C70100`

  – in executable as: `B700`

  Workaround:

  Define `var1` in the file where it is used.

  ```
      BYTES
      segment byte at ff 'ram0'
  .var1 DS.b 1
      WORDS
      segment byte at E000 'rom'
      LD {var1+1},A
  ```

- Semi-colon ";" only acts as a comment if it is preceded by a space.
- When `#define` is used twice with the same name but with different replacement strings, there is no warning and unexpected results may be obtained:

  ```
      #define CONST 5
      #define CONST 3
  ```

  is equivalent to:

  ```
      #define CONST 5
      #define 5 3
  ```

  And:

  ```
      ld a,#5
      ld a,#CONST
  ```

  is assembled as:

  ```
      ld a,#3
      ld a,#5
  ```

- When evaluating the difference between two labels located in different relative segments but defined in the same file, the generated code is wrong. The resulting list file, however, is correct.

Example:

For the following example, the code generated is **E701** instead of **E781**. The result in the list file is **E781 ld ({label2-ram0start},x),a**

st72334.asm:

```
segment byte at 80-FF 'ram0'
segment byte at 100-1FF 'stack'
```

eval.asm:

```
   BYTES
   segment 'ram0'
.ram0start
.var1 ds.b
   WORDS
segment 'stack'
.var2 ds.b
label2
   segment 'romlow'
   ld ({label2-ram0start},x),a
```

Three possible workarounds:

– Define the start addresses of the segments in the file where the difference is computed.

– Place the segment codes in two different files; however, the code is less optimized (for example D70081 instead of E781),

– Avoid computing the difference between labels located in two different segments. Get the same result by adding `$stackstartaddr-$ram0startaddr` (difference of constants) and `label2-stackstart` (difference within the same segment).

Example:

st72334.inc:

```
#define stackstartaddr 100
#define ram0startaddr 80
```

st72334.asm:

```
#include "st72334.inc"
segment byte at ram0startaddr-FF 'ram0'
segment byte at stackstartaddr-1FF 'stack'
```

eval.asm:

```
       #include "st72334.inc"
       BYTES
       segment 'ram0'
    .ram0start
    .var1 ds.b
       WORDS
       segment 'stack'
       stackstart
    .var2 ds.b
    label2
       segment 'romlow'ld ({{$stackstartaddr-
    $ram0startaddr}+{label2-stackstart}},x),a
```

- When the DATE directive is used immediately after a segment directive without declaring any constants, the generated code is wrong; the date is offset by 12 bytes.
- No automatic optimization for variables in page zero, which are not declared as BYTES or with a ".b" suffix:

  The programmer needs to specify that variables located in the zero page [0x00 - 0xFF] must be assembled with an 8-bit address; otherwise, these variables may be accessed as having 16-bit addresses with longer instructions. The programmer can request a short addressing mode either individually by using the ".b" suffix for a given variable, or globally by using the "BYTES" directive before the declaration of the variables in page zero. The "BYTES" declaration or ".b" suffix applies to the address of the variables and not to their contents. It is thus independent of the size of the variables:

```
       BYTES
       segment byte at 80-FF 'zero page ram'
    var1: ds.b 1
    var2: ds.b 2
    var3: ds.b 1
```

  or

```
       segment byte at 80-FF 'zero page ram'
    var1.b: ds.b 1
    var2.b: ds.b 2
    var3.b: ds.b 1
```

- Start delimiters for `#DEFINE` also include base specifying prefixes such as: `$, &, %, ~` (Motorola format) leading to the surprising result with the following piece of code:

```
    #define FE 1
    ld A,#$FE
```

  `ld A,#$FE` is assembled as `ld A,#$1`.
- No blank is allowed between "," and "`X`" inside `( )`.

  `LD A,($100,X)` is accepted.

  `LD A,($100, X)` is not accepted.
- No blank is allowed between "," and "`Y`" inside `( )`.

  `LD A,($100,Y)` is accepted.

  `LD A,($100, Y)` is not accepted.

4 Release information for previous releases

4.1 ASM assembler release history

ASM release 4.52

There is a fix for the generation of list files with absolute addresses by a second pass of the assembler after linking, in order to have correct debug information for gdb7:

```
asm -li <file1>.asm -fi=<application>.map
```

The maximum size of a line read in the map file line has been extended from 256 to 512 characters. Otherwise, when the application path name is too long, the following warning occurs: "as1 : Warning 73: Couldn't find entry for segment '<segment name>' in Mapfile '<application>.map'".

LI option has been modified: it is now possible to specify the extension of the list file. It does not have to be LST as before. The extension must have three alphanumeric characters. The default extension is LST.

Two bugs have been fixed in the list file with relative addresses:

1. Fix for number of bytes of labels: Add condition to the update of the number of bytes of the previous label. It must be updated if the two conditions are met: it has the same address and it is also in the same segment as the current label.
2. Fix for line number of labels defined in include files.

Fix for name of source file in list file header: it is now always the name of the assembled file and never the name of the file being included.

Always print file names with lower case letters in the list file and in the map file.

Fix for address size for negative numbers in symbol table printed in list file.

Fix for line 2 of list file in the case where there is a comment after the end directive.

Assembler release 4.51

If there is no argument after any of the following directives:

- dc.b
- dc.w
- dc.l
- byte
- word
- long

then a warning message is generated: "No argument after <directive>".

Otherwise, the behavior is unchanged: the generated code is the same.

Assembler release 4.50

Corrected: Extra list file optimization suppressed. To resolve the known problem where instructions, with operands involving a computation on a label in page 0 resulting in an address outside page 0, may be incorrectly assembled in the absolute list file.

Assembler release 4.49

- Support for porting to Linux
- Corrects the banner output in list file

ASM release 4.48

Corrected:

- Only one error message is generated when a source file does not exist.
- The following improvements have been implemented for the `Define` (`-D`) option:
  - When specifying the second argument, the use of **=** is equivalent to using a **blank space**. For example, `-D <string x>=2` is equivalent to `-D <string x> 2`.
  - It is possible to use only one argument. For example, **`-D <string x>`**. In the absence of a second argument, `<string x>` is replaced with `1`. The implementation of **`-D <string x> -D <string y>`** has been corrected so that both `<string x>` and `<string y>` are replaced with `1`.
- When using the Include (-I) option, the notation `-I=dir1 -I=dir2` is now equivalent to using `-I=dir1;dir2` when specifying multiple include paths.
- Using a label with an unknown suffix results in an `Error 89: Illegal suffix`. Recognized suffixes are: `.b, .w, .l`.

ASM release 4.47

New features: Environment variable `METAI`, used to find the file containing the ST7 instruction set (*st7.tab*), is no longer taken into account. The assembler searches for *st7.tab*, first in the current directory, then in the directory where the assembler executable is located.

Corrected:

- Generation of build dependencies using `-M` has been fixed. Warnings and errors are no longer displayed. If an error occurs, no build dependencies are displayed. Files that are included more than once appear only once in the dependencies.
- The “file truncated” error has been suppressed. In previous versions, a false error was generated when the assembler accessed files with filenames that exceeded 8 characters.

ASM release 4.46

Corrected: This release corrects two regression problems related to using the short addressing mode and the `-fi` option. In the previous version, the use of a difference as an operand of an instruction with short addressing mode caused an assembly error in the two following cases:

- When the difference was defined in a relative segment using the `EQU` or `CEQU` directive, and the difference was between the current address symbol (`*`) and a non-public label in the segment.
- When the difference was defined in a relative segment using the `EQU`, `CEQU` or `#DEFINE` directive, and the difference was between two labels defined in the segment, but where only one label had been declared public outside the segment.

ASM release 4.45

The assembler has been extended to allow 4096 `#define` statements. Depending on the length of `#define` statements, users may reach this limit sooner.

ASM release 4.44

Corrected:

- Incorrect display of assembly source lines in STVD7 source window due to extra list file optimization. The code listed in the absolute list file is now the same as the code generated by the linker.
- Incorrect display of assembly source lines in STVD7 source window due to excessive line length. Lines exceeding 255 characters are no longer accepted.

ASM release 4.43

Corrected: Incorrect display of assembly source lines in STVD7 source window due to extra list file optimization: the correction was partial.

ASM release 4.42

Corrected: For the option **-LI** under a UNIX environment, the list file generated now includes macro calls before their expanded codes (the same as when operating in a Windows environment).

ASM release 4.41

Corrected:

- When using option **-M**, if a file path name includes white spaces, it is printed between double quotes in the list of dependencies.
- In error messages, the full path name of a source file is displayed.

ASM release 4.40

New options:

- `-I` where `-I =`“`<path1>;<path2>;...;<pathN>`” This option specifies the list of search paths for files that are included or loaded. Each path is separated by a semicolon (;). The path list must be enclosed within double quotes.
- `-M` This option obtains the list of dependencies (for example, files opened during the build process such as source files, included files and loaded files).

Modified options: `-LI` where -LI = <listing file path name>

It is now possible to specify the full path name of the listing file. Note that the file extension cannot be changed—it is always “.lst”.

Corrected: `#LOAD` directive: A warning is issued when a text file is loaded. Only purely binary files should be loaded. Files in MOTOROLA or INTEL text format are not binary files.

ASM release 4.30

New directives (only useful when debugging with STVD7):

- `NEAR <string>` This directive is used with functions called by CALL or CALLR, allowing the debugger to correctly search the stack for the return address of the calling function.
- `INTERRUPT <string>` This directive is used with interrupt handlers and so aids the debugger in correctly searching the stack for the return address of the interrupted function. More specifically, this directive allows the debugger to distinguish between a call (which causes two bytes to be written to the stack) and an interrupt (which causes five bytes to be written to the stack).

ASM release 4.20

Corrected:

- When compiling large assembly files previously, an error "`47: Out of label space`" would occur. The bug has been fixed.
- The progress bar (star characters) used to signify that assembly is in progress has been suppressed as it generated strange results when using the assembler with certain IDE tools.

ASM release 4.10

Corrected: A problem existed where the `-D` option did not work—the second argument was not taken into account. The `-D` option format is as follows:
`-D <first argument> <blank> <second argument>`
The option works now, provided that you respect the following rule: if the `-D` option has no second argument, `-D` must be placed at the end of the line.

ASM release 4.01

Corrected: Empty segment generated unsuitable error message. Now generates a warning:

```
"Warning 73: Couldn't find entry for segment '<segment name> in
mapfile <file.map>"
```

ASM release 4.00

New features:

- Port Assembler is now a 32-bit application under Windows® 95 and Windows® NT®. Windows® 3.11 is no longer supported.
- "`-K`" option, which allows to generate the same number of lines as the source file.

ASM release 3.00

Note: ***Important information****: In order to improve the debugging information reported by the linker LYN, the object code (*.obj) has been changed in version 2.0. As a result, when using this package, always recompile all your assembly files, if your assembler has a version earlier than 2.0. Not doing so will produce the following message:* *`lyn : Error 7: Corrupted Object File`*

New features: The assembler now supports files larger than 64K characters. The upper limit is 10 Mbytes.

Corrected: ASM with `-fi` option could produce a listing format with wrong start and end addresses, when the directory path name contained dot character(s) '.', (91).

Minor problems: When using the `-fi=<file>.map` option, the assembler may fail when:

- Some segments are empty (error 73). The workaround is to comment them out.
- You try to assemble a file that has not been used for producing; the *.map* file error 73 occurs. The workaround is to use the file during the first link.
- Some EXTERN labels are never used (warning 80). The workaround is to comment them out.

Warnings: The Crash-Barrier assembler ASM displays unexpected behavior, when a `#define` is used twice with the same name, but with different values, for example:

```
#define OP1 1
...
#define OP1 4
```

ASM replaces OP1 by 1 in second `#define`, then replaces all occurrences of 1 by 4.

4.2 LYN linker release history

Linker release 3.19

- When using -no_overlap_error option, the overlap message size limit has been extended from 256 to 2048 characters.

Linker release 3.18

- The *.cod* executable file is removed for most linker errors (since Lyn 3.17).
- Segment overlaps are detected and an error message is generated (since Lyn 3.17). The executable is no longer generated. This is the default behavior.
- Segment overlaps are detected and a warning message is generated if `no_overlap_error` option is used but the executable is generated.
  `lyn -no_overlap_error "file1.obj+file2.obj,appli.cod;"`
- The application size limit has been extended from 192 kilobytes to 1024 kilobytes.

Linker release 3.16

Adds support of command lines longer than 512 characters (size up to 4096 characters).

LYN release 3.14

Maximum length for full path names is extended to eliminate an error that occurred.

LYN release 3.13

The “file truncated” error has been suppressed. In previous versions, a false error was generated when the assembler accessed files with filenames that exceeded 8 characters.

LYN release 3.12

Corrected:

- The generation of multiple output files now works correctly under Windows.
- A regression introduced in LYN release 3.11 has been fixed.

LYN release 3.11

The maximum size of path names in map files has been increased to up to 1000 characters.

LYN release 3.10

Minor bug fix related to 32-bit compatibility.

LYN release 3.00

New features: Port Linker is now a 32-bit application under Windows® 95 and Windows® NT®. Windows® 3.11 is no longer supported.

Corrected: Progress bar suppressed—it generated strange characters on `stderr`, especially when using the assembler with an IDE tool.

4.3 Obsend release history

Obsend release 2.14

The application size limit has been extended from 192 kilobytes to 1024 kilobytes.

Obsend release 2.13

Corrected: `ix` output format option correctly generates the Intel Hex-32 type 04 record.

Obsend release 2.12

Corrected: There are no more problems with full path names.

Obsend release 2.11

Corrected: End s-record S8 has now a 3-byte address.

Obsend release 2.10

Internal release only.

Obsend release 2.00

New features: Port Obsend is now a 32-bit application under Windows® 95 and Windows® NT®. Windows® 3.11 is no longer supported.

4.4 Librarian (LIB) release history

Librarian release 2.02

Support for porting to Linux.

Librarian (LIB) release 2.0.1

Correction of banner display.

Librarian (LIB) release 2.0

LIB is now a 32-bit application under Windows® 95 and Windows® NT®.
Windows® 3.11 is no longer supported.

4.5 MSCI post processor release history

MSCI post processor release 1.05

Recognizes old and new banner format in list files.

MSCI post processor (st7msci2) release 1.03

Initial release as part of the ST7 Assembler-Linker toolset.

The installation of the ST7 Assembler-Linker includes an additional program (*st7msci2.exe* v.1.03) and a language description file (*msci.tab* v.1.01), designed for the assembly of routines for the ST7267 Mass Storage Communication Interface (MSCI) peripheral, written in Assembly language.

Note: These two components are ***not*** *used by the ST7 Assembler-Linker when assembling ST7 applications. These are* ***only*** *used when assembling routines for the ST7267 MSCI peripheral.*

These components are designed for use with this version of the ST7 Assembler-Linker toolset (ASM 4.48, LYN 3.14). For further information, refer to the *ST7267 Datasheet* and the *ST7 MSCI Assembler User Manual*.

4.6 st7.tab release history

st7.tab release 1.3

Opcodes `JRH` and `JRNH` that were erroneous are now fixed.

4.7 stm8.tab release history

stm8.tab release 1.2

Support for STM8 microcontrollers is provided with the *stm8.tab* file, version 1.2.

4.8 ABSLIST release history

ABSLIST release 1.01

Accepts map files with lines which have more than 256 characters. The new size limit is 2048 characters.

ABSLIST release 1.00

ABSLIST is a post processor which reads a list file with relative addresses and unresolved symbols and converts it into a list file with absolute addresses and resolved symbols. The list file with relative addresses is first generated by the assembler. For the conversion, the post processor needs information which is located in two files generated by the linker: the map file and the executable file. As it uses the data output by the linker, ABSLIST is necessarily in line with the executable code, which was not the case with the assembler when it was called a second time after linking the application.

Example:

```
ABSLIST file1.lsr -o file1.lst -exe appli.s19 -map appli.map
```

(produces *file1.lst* from *file1.lsr*)

`-o and -map` options may be omitted. The preceding command can be reduced to:

```
ABSLIST file1.lsr -exe appli.s19
```

The full syntax is described in the ST Assembler-Linker manual.

Respect the following constraints:

- Do not use .NOLIST, .XALL and .SALL primitives so as not to hide code where segments are declared or labels are defined.

Define as public the labels that are set by EQU or CEQU to an expression involving a label defined in a relative segment. ABLIST is not able to resolve the addresses if such a label is private.

5 Revision history

Table 1. Document revision history

Date	Revision	Changes
14-Jun-2006	1	Release for update of ASM 4.49, LYN 3.16, LIB 2.02, MSCI 1.05.
22-May-2008	2	Update for releases of `stm8.tab` version 1.2 and LYN 3.17.
03-Jun-2008	3	Update for releases of LYN 3.18 and Obsend 2.14.
22-Sep-2008	4	Update for release of ASM 4.50.
18-May-2009	5	Update for release of ASM 4.51.
20-Nov-2009	6	Update for release of ASM 4.52 and ABSLIST 1.00.
03-Oct-2010	7	Update for release of LYN 3.19 and ABSLIST 1.01.
16-Jan-2012	8	Update for release of OBSEND 2.15.