APPLICATION NOTE

STARTING WITH ST7 ASSEMBLY TOOL CHAIN

by the 8-Bit Micro Application Team

INTRODUCTION

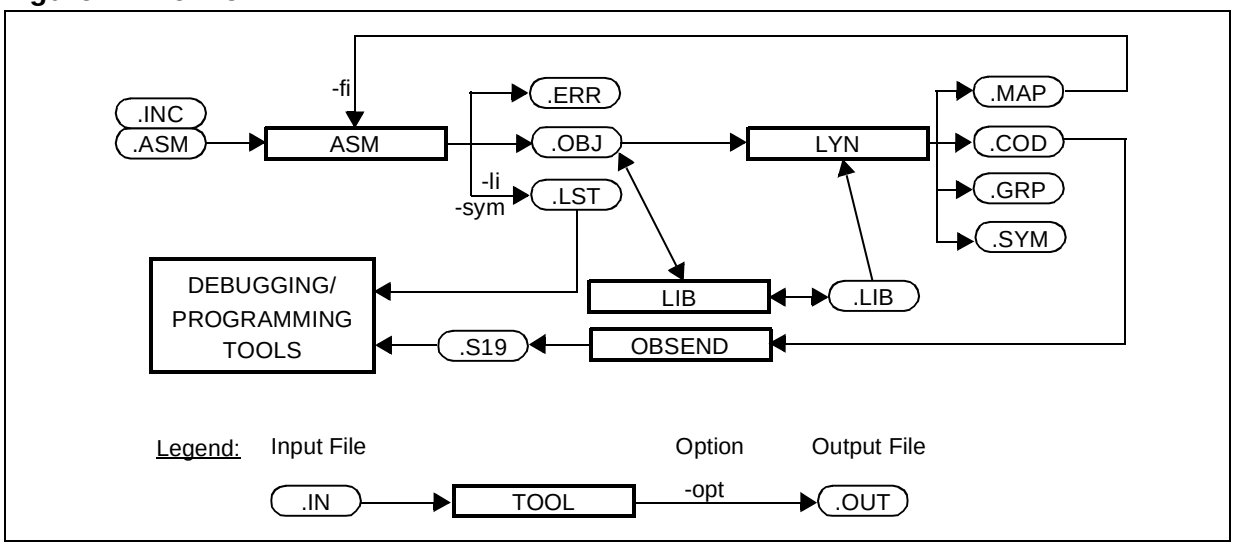
The purpose of this document is to give guidelines for starting a ST7 application design based on the STMicroelectronics Assembly tool chain.

1 ST7 TOOL CHAIN OVERVIEW

The ST7 tool chain is a cross development system for ST7 microcontroller based applications. It runs in the DOS environment on a PC machine. The UNIX environment will not be described in this note; please refer to the user documentation for this purpose (ST7 Software Tools User Manual).

A first overview is given in Figure 1. Assembler options are described in chapter 5.3.

Figure 1. File Flow



It includes the following development tools:

- ASM: Assembler for the ST7 microcontroller.
- LYN: Linker for resolving cross-references and allocating memory to object files.
- OBSEND: File formatter for generating the S19 files for the debugging tools.
- LIB: Librarian utility for creating and maintaining object file libraries.

2 INSTALLATION PROCEDURE

The ST7 Assembly Tool Chain can be installed from Internet or from the ST7 CD ROM. It's completely free of charge and you can easily download the last version of this software from the ST Internet site (http://st7.st.com).

When the installation is finished, do not forget to reboot your PC because the `autoexec.bat` file is automatically modified as follows:

```
PATH=c:<installation_directory>
SET METAI=c:<installation_directory>
SET DOS4G=QUIET
```

The second statement allows the assembler to find the `ST7.TAB` file in the installation directory even if the assembler is executed from your development directory.

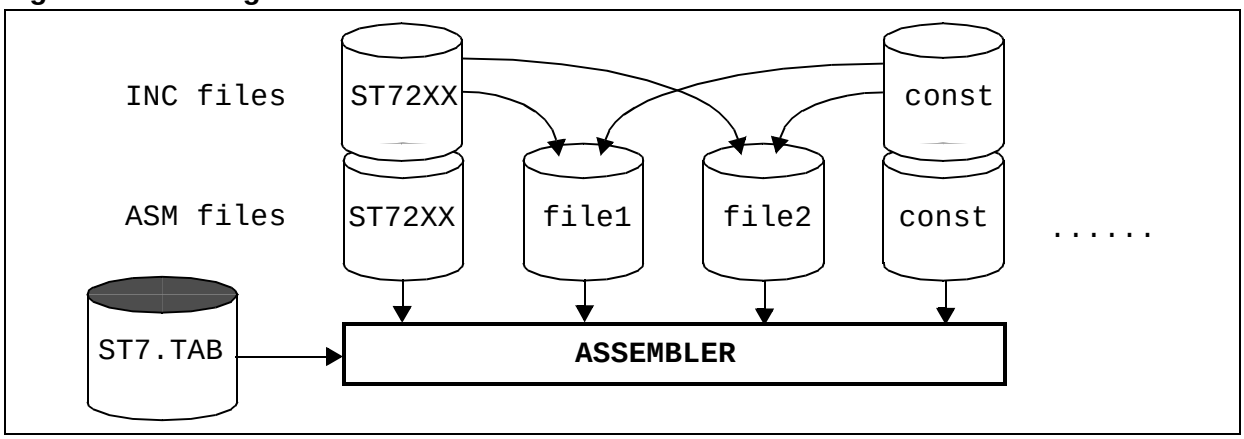
The last one includes the installation directory in your DOS path so that you can launch the tools from your development directory.

3 SETTING UP A NEW APPLICATION

3.1 ST7 SOFTWARE LIBRARY

The ST7 software library is formatted as described in Figure 2.. To be fully compatible with the ST standard software library and support, you are strongly advised to use the same file organization . However, for very simple applications, the complete source code may be written in a single source file.

Figure 2. Files organisation



The ST7 software library is available on ST Internet web site.

Default file type definition:

ST7.TAB	ST7 description table provided with the tool chain.
ST72XX.asm	Hardware register definition (see software library).
ST72XX.inc	Register prototypes to be included in all modules.
const.asm	Constants and global variables definition.
const.inc	Constants and global variables prototypes.
fileX.asm	Modules containing the code of your application.

4 ASSEMBLER DESCRIPTION

4.1 ASSEMBLER SYNTAX

4.1.1 Labels, mnemonics and directives

A label can be a variable name or an address in your code which is used to simplify the memory access and JUMP instructions. Labels should always start in the first column; conversely, every statement starting in the first column will be considered as a label.

Mnemonics are given names to simplify the use of opcodes. They should never be put in the first column. Please refer to the ST7 Programming Manual for their complete description.

Directives give specifications to the assembler or linker to modify their process. They should never be put in the first column either. Please refer to the ST7 Software Tools User Manual for their complete description.

Figure 3. Assembly File Example (fileX.asm)

```
            #INCLUDE "ST72251.INC"; Assembler Directive

            WORDS                  ; Assembler Directive
            segment 'rom'          ; Linker Directive
            ...
        .NEXT                      ; Label
            LD (Table, X), A       ; Mnemonic
            DEC X                  ; Mnemonic
            JRPL NEXT              ; Mnemonic
            ...
```

Note: the semicolon stands for the comment delimiter. It is valid from its position for the rest of the current line.

4.1.2 ST7/ and END statements

All assembly source files must start with the ST7/ statement in the first row, first column (this the only exception to the rule stated in paragraph 4.1.1). This will tell the assembler that the target processor is an ST7 i.e. the description table can be found in the ST7.TAB file.

All assembly source files must finish with the END statement. Forgetting this statement will generate an error. Take care to put a carriage return after the END statement to be sure it will be taken into account during the assembly.

Note: Be aware that any code put after the END statement will not be assembled; for this reason, never put an END at the end of an include file otherwise the code in the ASM file including that INC file will never be assembled.

4.1.3 Defining constants and variables

RAM space may be reserved to store variables using the ds.b and ds.w statements (ds.b for bytes and ds.w for words, see Figure 4.). ROM space may be reserved to store constants using the dc.b (or BYTE) and dc.w (or WORD) statements. The content of the reserved memory follows these directives (see Figure 4.). Using WORD instead of dc.w will reserve memory with the least significant byte first. BYTE and dc.b are exactly the same.

Note: use the ds. statement whenever you want to allocate memory to a variable, instead of using the EQU statement which does not reserve any memory space.

Figure 4. Variables and Constants definition (const.asm file)

```
    BYTES
    segment 'ram0'
.count ds.w 1           ; reserve a word in page 0 for count
    WORDS
    segment 'ram1'
.step ds.b  3           ; reserve three bytes in ram for step
    segment 'rom'
.rate dc.w 9600         ; reserve a constant word in rom
.tab dc.b $AA,%01010101 ; reserve two constant bytes in rom
```

4.1.4 BYTES, WORDS and segment definition

The two assembler directives BYTES and WORDS specify if the labels subsequently defined have a 8 bit or 16 bit address. Therefore, BYTES should be used before defining all hardware registers and RAM variables in page zero (before address $FF). WORDS should be used in any other case i.e. for RAM variables in other pages, EEPROM variables, code, constants and interrupt vectors in ROM.

The segment linker directive is used to define a memory range. This is done once and for all in the ST72XX.ASM file as it is only dependent on the ST7 microcontroller you are using. This directive is also recalled every time you want to put variables or code in a different range.

Note 1: The assembler is not able to find the label length from the reserved word segment which is only understood by the linker. That is why you have to specify the label length with BYTES or WORDS even if it sometimes seems redundant.

Note 2: WORDS is the default statement. You can put it at the beginning of your ASM files and at the end of your INC files.

Note 3: BYTES and WORDS do not specify the variable length but the variable address length. You may have a word (16-bit variable) in page zero and a byte (8-bit variable) stored after address $100. This means you must not mix the BYTES and WORDS directives with BYTE and WORD directives.

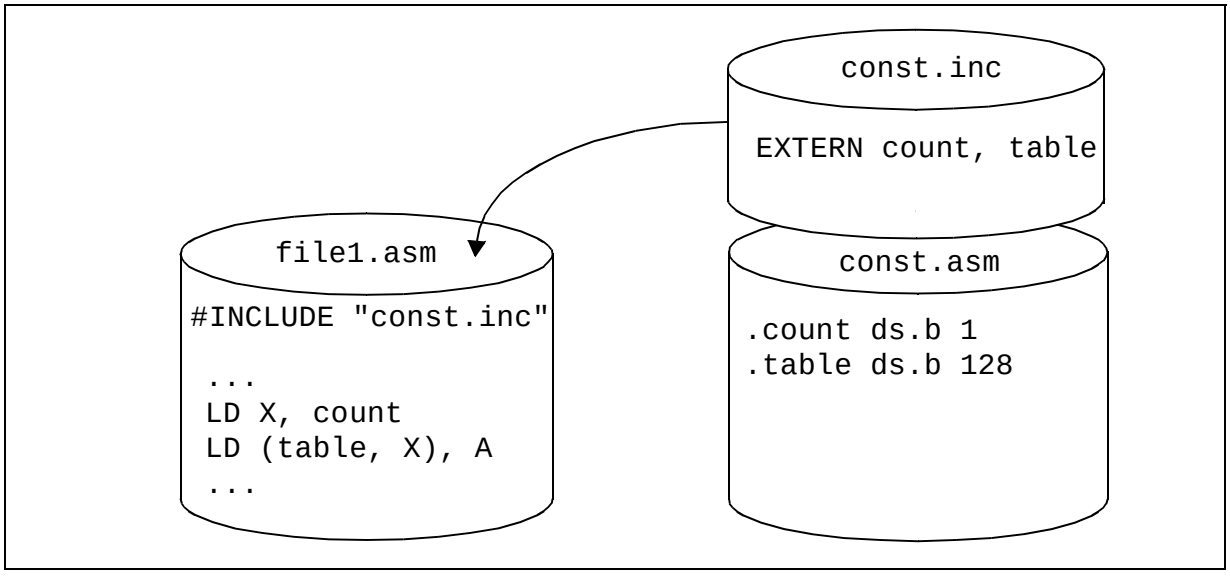
4.2 FILE MANAGEMENT

4.2.1 PUBLIC, LOCAL and EXTERN references

By default, all labels are local to one file. If you want to use a label in another module, you have to specify it as global. This can be done either by putting a dot before the label or using the PUBLIC directive at the beginning of the file followed by the list of global labels. In the other module, use the directive EXTERN followed by the list of external variables you want to use (refer to Figure 5.).

Note: local labels are lost at assembly stage. If you want to see all symbolic information in the debugger windows, define all your label as *PUBLIC* even if they are not used in another module.

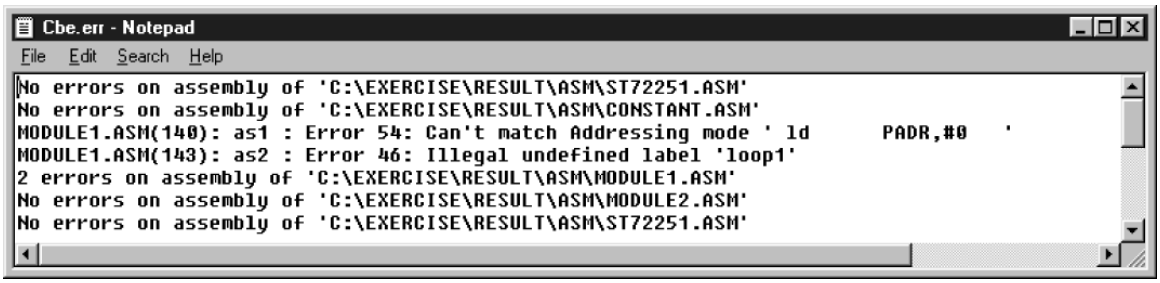
Figure 5. Using Public and External labels



4.2.2 Error file

By default, an error file cbe.err is generated by the assembler and placed in the current directory. This file specifies the number of errors encountered, the file name and line number as

well as the description of each error. This file makes it easier to debug the code and may be parsed by a integrated editor.



5 OTHER TOOLS

5.1 LINKER

The linker has no specific option or configuration file. All information to the linker is given through specific directives in the source file such as the `segment` directive. To invoke the linker `LYN`, the command line has to specify:

– the list of object files to link together separated by a +

– the name of the `COD` file you want to generate

– the list of libraries which may be used.

All this information may be written in the command line or put in a response `RSP` file as specified in Figure 6.

Figure 6. Invoking the linker

c:>LYN ST72251+const+file1+file2, appli, library
or
c:>LYN @LIST.RSP
if the RSP file content is:

```
ST72251+const+file1+file2
appli
library
```

5.2 OBSEND

OBSEND is a general purpose tool used to convert the `COD` file generated by the linker into an `S19` record file which is used by the debugging and programming tools. The command line (see Figure 7.) has to specify:

– the `COD` file output by the linker

– the destination type which can be the screen (`v` for video) or a file (`f`)

– the name of the `S19` file you want to generate (if the destination is a file)

– the format which must be s or x for an S19 record format (refer to the documentation to have the other available formats).

Figure 7. Invoking OBSEND

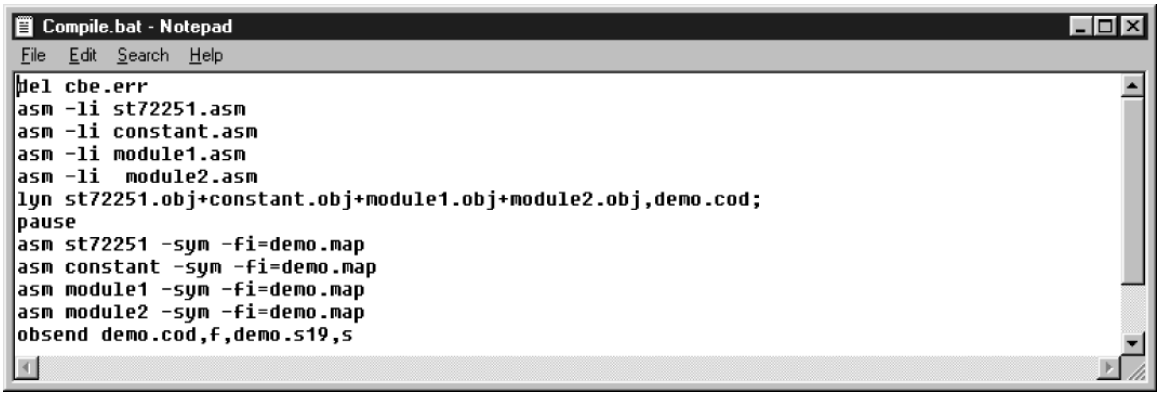
```
c:>OBSEND appli, f, code.S19, s
```

Note: the command line has to specify the S19 extension as it is HEX by default.

5.3 DEBUGGING THE CODE

Once all the assembly errors have been removed, the code needs to be debugged with an emulator. To be able to access all the symbolic information (variable names and labels) from the debugger, the ASM files need to be assembled a second time to update the listing file with the symbolic information held in the map file. This is performed by the `-sym` assembler option used in conjunction with the `-fi` assembler option.

As the command lines will have to be typed in several times, a batch file can be written to automate the process:



Note 1: the first time, the assembler does not know the absolute addresses because they are computed by the linker (refer to Figure 1.). A look at the listing file generated by the `-li` assembler option will show zeros for all absolute addresses.

Note 2: a semicolon can be put at the end of the linker command line to specify there is no other argument (no library). If you omit this semicolon, you will be prompted for libraries every time you launch your batch file.

"THE PRESENT NOTE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH A NOTE AND/OR THE USE MADE BY CUSTOMERS OF THE INFORMATION CONTAINED HEREIN IN CONNEXION WITH THEIR PRODUCTS."

littlediode
COMPONENTS FOR ALL

Littlediode supplies new, hard to find or obsolete electronic components and semiconductors all over the world.

With over two million different components listed you are sure to find the part you need.

Feel free to visit us today at our online store:

LittleDiode.com

Looking forward to providing you with the best possible service.