ST7 MICROCONTROLLER TRAINING

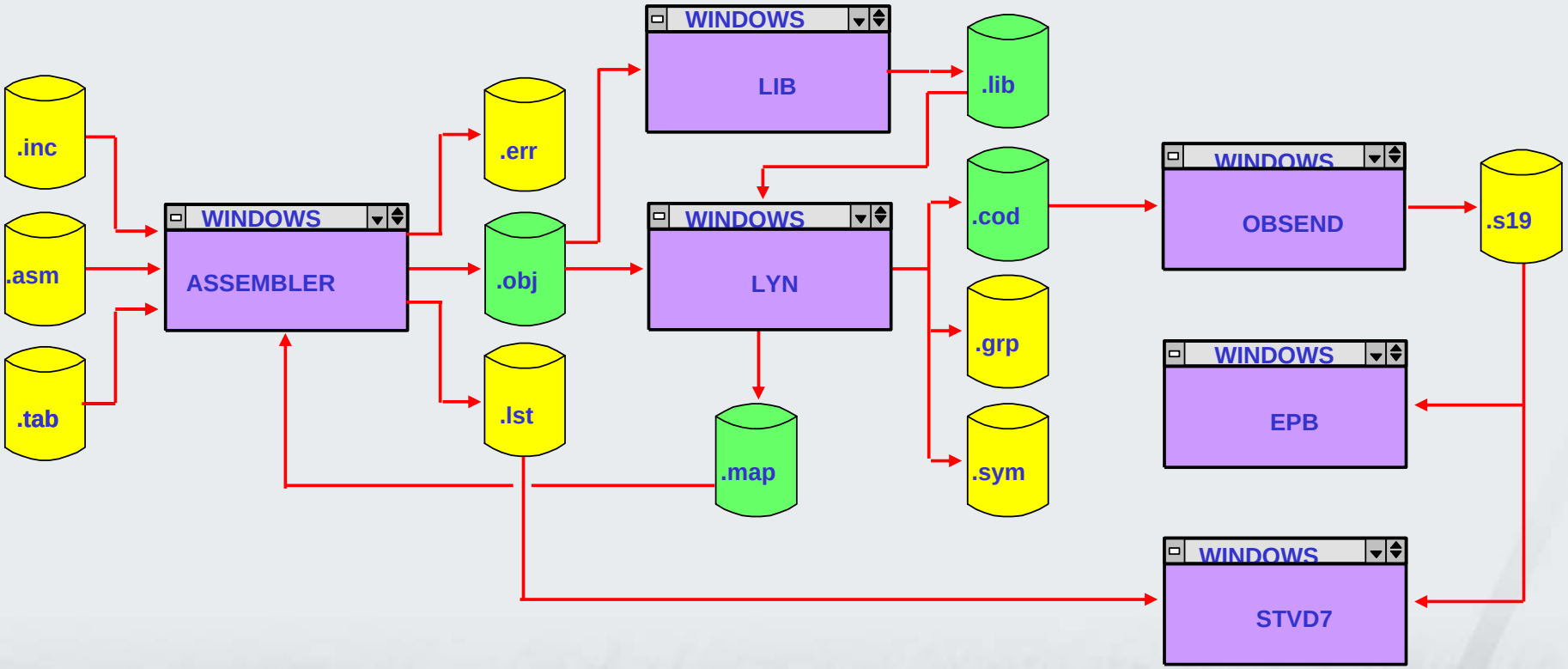
1. INTRODUCTION
2. CORE
3. ADDRESSING MODES
4. ASSEMBLY TOOLCHAIN
5. STVD7 DEBUGGER
6. HARDWARE TOOLS
7. PERIPHERALS
8. ST-REALIZER II
9. C TOOLCHAINS

SOFTWARE TOOLS OVERVIEW

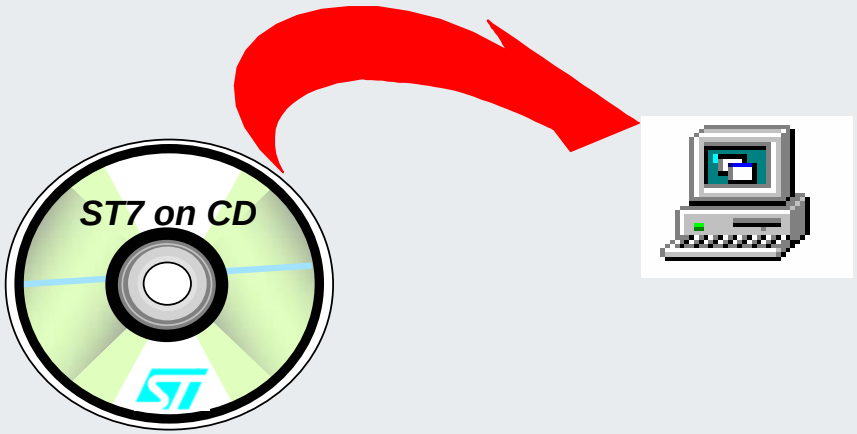
- STMicroelectronics Toolchain
  - **Cross assembleur: ASM**
  - **Linker: LYN**
  - **Formatter: OBSEND**
  - **Librarian: LIB**
  - **Windows Debugger: STVD7 (IDE)**
- Raisonance:
  - **Soon available (Q2 2006)**
- Cosmic C Toolchain
  - **Cross assembleur: CAST7**
  - **C Compiler: CXST7**
  - **Windows Debugger: ZAP**
  - **IDE: IDEA**

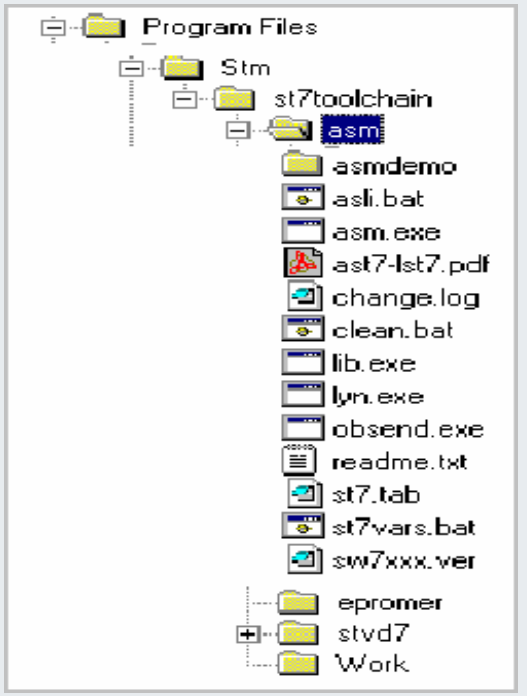
RAISONANCE

ST7 SOFTWARE TOOLS
Assembly toolchain



STM TOOLCHAIN OVERVIEW





- THE AUTOEXEC.BAT (THE WINDOWS START UP FILE) is automatically modified at the end of the installation (adds of paths):
  - **C:\Program Files\Stm\St7toolchain\ASM**
    set METAI=C:\Program Files\Stm\St7toolchain\ASM
    set DOS4G=QUIET

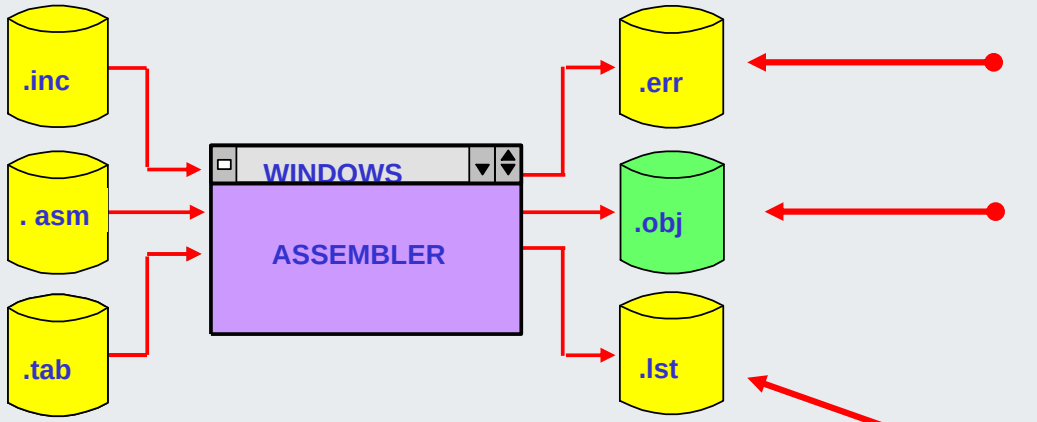
THE ASSEMBLER: ASM
Overview (1)

- The ASM assembler produces code for a target machine

- To target the ST7:
  - **The first line of the source code is reserved for specifying the target processor:**
    - ✓ **ST7/ ; the first line is reserved for specifying the ; instruction set of the target processor**

  - **A machine description file is installed in C:\Program Files\STM\St7toolchain\asm: "ST7.TAB".**
    - ✓ **This file describes the different opcodes used by the ST7 assembler.**

THE ASSEMBLER: ASM
Overview(2)



Error file: contains fatal and recoverable errors

Object file (will be used by the linker)

Listing file

ASM <file to assemble>, <listing file>, <switches>

THE ASSEMBLER: ASM
Options...

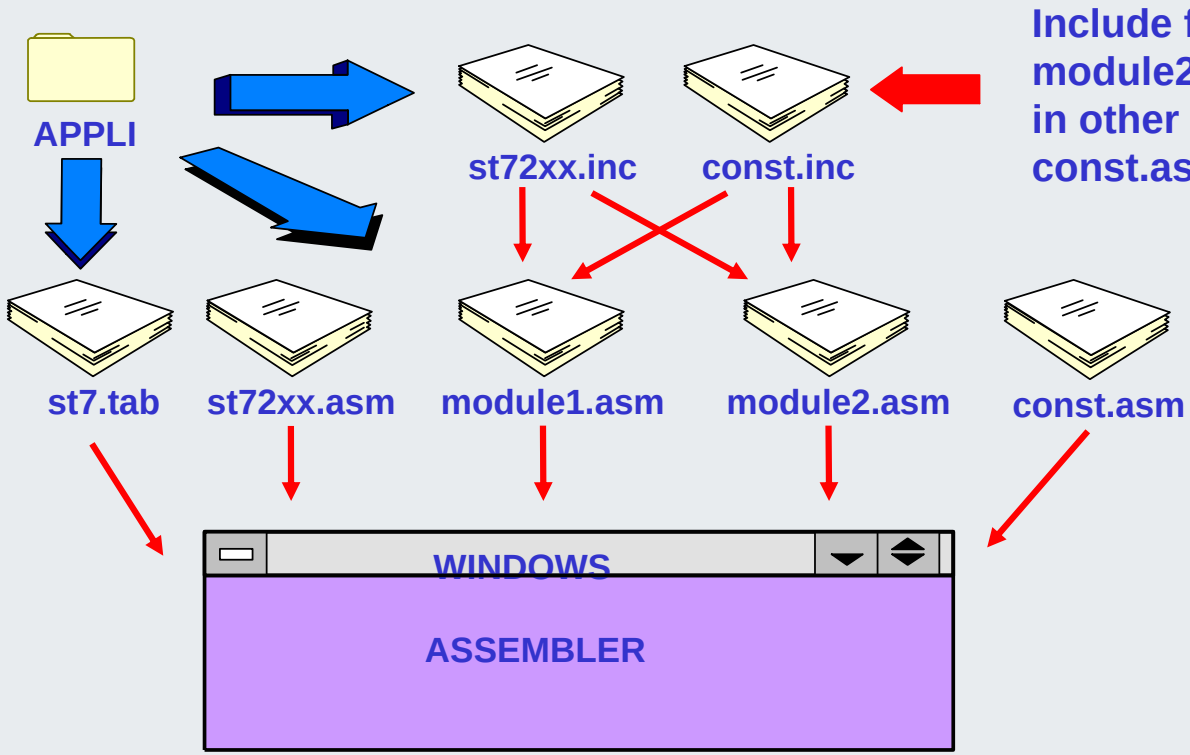
- Options:

-LI	enable the listing generation
-SYM	enable the symbol table generation
-OBJ=<path	specifies the object file path
-FI=<mapfile	updates the symbol table in the listing file
-D<1 <2	defines 'string1' to 'string2'
-PA	enable pass-1 listing
-NP	disable error generation

- Example:

asm appli.asm -Li -D EPROM 1 -D RAM 2
will generate the listing file appli.lst and replace eprom by 1 and ram by 2

THE ASSEMBLER: ASM
Program organisation



Include files: module1 and module2 can refer to labels defined in other modules (for instance const.asm)

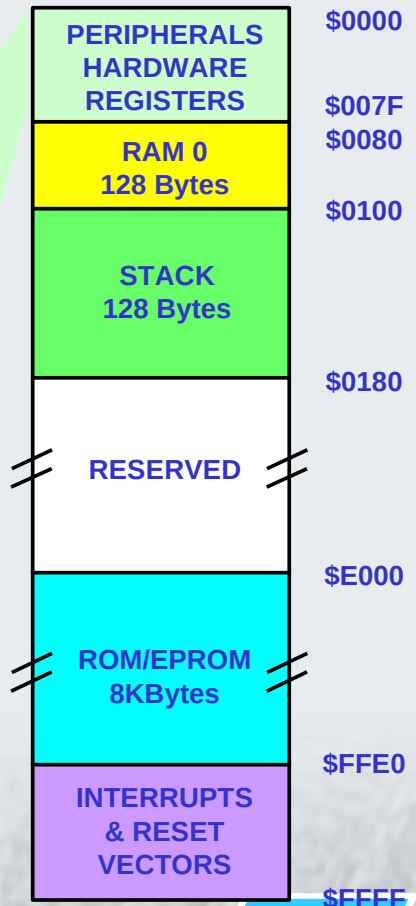
- st72xx.asm: Register and memory mapping definition
- const.asm: constant definition
- st7.tab: machine description file

EXAMPLE
ST72254.asm (1)

st7/

To target the ST7

```
;*******************************************************************
;* ST72254.ASM                              date: 01/05/2000    *
;* Device : ST72254                                             *
;*******************************************************************
   BYTES         ; following addresses are 8 bit length

;*******************************************************************
   segment byte at 0-7F 'periph'
;*******************************************************************

;*******************************************************************
;*                  I/O Ports registers                            *
;*******************************************************************
.PCDR       DS.B   1      ; port C data register
.PCDDR      DS.B   1      ; port C data direction register
.PCOR       DS.B   1      ; port C option register
            DS.B   1      ; not used
.PBDR       DS.B   1      ; port B data register
.PBDDR      DS.B   1      ; port B data direction register
....
```



EXAMPLE
ST72254.asm (2)

```
;*******************************************************************
   segment byte at 80-FF 'ram0'
;*******************************************************************

WORDS          ; following addresses are 16 bit length

;*******************************************************************
   segment byte at 100-17F 'stack'
;*******************************************************************

;*******************************************************************
   segment byte at E000-FFDF 'rom'
;*******************************************************************

;*******************************************************************
   segment byte at FFE0-FFFF 'vectit'
;*******************************************************************
      END
```

EXAMPLE
Module1.asm (1)

st7/

First line, first character !!!

```
; the first line is reserved
; for specifying the instruction set
; of the target processor
```

```
;*******************************************************************
;* module1.ASM                                date: 01/05/2000    *
;* Device : ST72254                                               *
;*******************************************************************
```

Title will be included on the first line of each page in the listing file

```
TITLE    "module1.ASM"

MOTOROLA                    ; format for the assembly (default)

#INCLUDE "st72254.inc"      ; include st72254 registers and memory mapping file
#INCLUDE "const.inc"        ; include general constants file
```

Motorola format (default one)
Hex: $A3 Bin: %10100011
Oct: ~243 Current PC: *

EXAMPLE
Module1.asm (2)

```
;*********************************************************************
;*     Program code                                                  *
;*********************************************************************

      WORDS

   segment 'rom'

.main
      ld     A,#$01          ; Initialization
       ...


;*********************************************************************
;*     Main program                                                  *
;*********************************************************************

.loop
     ....
      jp     loop            ; Make another conversion
```

Segment called 'rom': beginning of the user code (cf. ST72XX.INC)

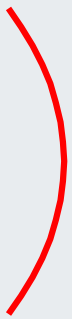
Initialization

Public label (definition starting with a "dot" character)

Main loop

EXAMPLE
Module1.asm (3)

```
;*****************************************************************
;* INTERRUPT SUB-ROUTINES LIBRARY SECTION                        *
;*****************************************************************
sw_rt       iret
ext0_rt     iret
ext1_rt     iret
css_rt      iret
spi_rt      iret
tima_rt     iret
timb_rt     iret
i2c_rt      iret
dummy       iret

;*****************************************************************
;* SUB-ROUTINES LIBRARY SECTION                                  *
;*****************************************************************
```



No interrupt subroutine in this example

Regular subroutine

```
.sub
            ...
            ret
```

EXAMPLE
Module1.asm (4)

```
;*****************************************************************
;* INTERRUPT VECTOR SECTION                                      *
;*****************************************************************
      segment 'vectit'

            DC.W    dummy           ;FFE0-FFE1h location
            DC.W    dummy           ;FFE2-FFE3h location
.i2c_it     DC.W    i2c_rt          ;FFE4-FFE5h location
            DC.W    dummy           ;FFE6-FFE7h location
            DC.W    dummy           ;FFE8-FFE9h location
            DC.W    dummy           ;FFEA-FFEBh location
            DC.W    dummy           ;FFEC-FFEDh location
.timb_it    DC.W    timb_rt         ;FFEE-FFEFh location
            DC.W    dummy           ;FFF0-FFF1h location
.tima_it    DC.W    tima_rt         ;FFF2-FFF3h location
.spi_it     DC.W    spi_rt          ;FFF4-FFF5h location
.css_it     DC.W    css_rt          ;FFF6-FFF7h location
.ext1_it    DC.W    ext1_rt         ;FFF8-FFF9h location
.ext0_it    DC.W    ext0_rt         ;FFFA-FFFBh location
.softit     DC.W    sw_rt           ;FFFC-FFFDh location
.reset      DC.W    main            ;FFFE-FFFFh location

END
```

Segment called 'vectit': interrupt vectors (start address: FFE0)

Define word in object code

End of source code (only for asm files).
The "end" directive must be followed by a Carriage Return

EXAMPLE
Const.asm (1)

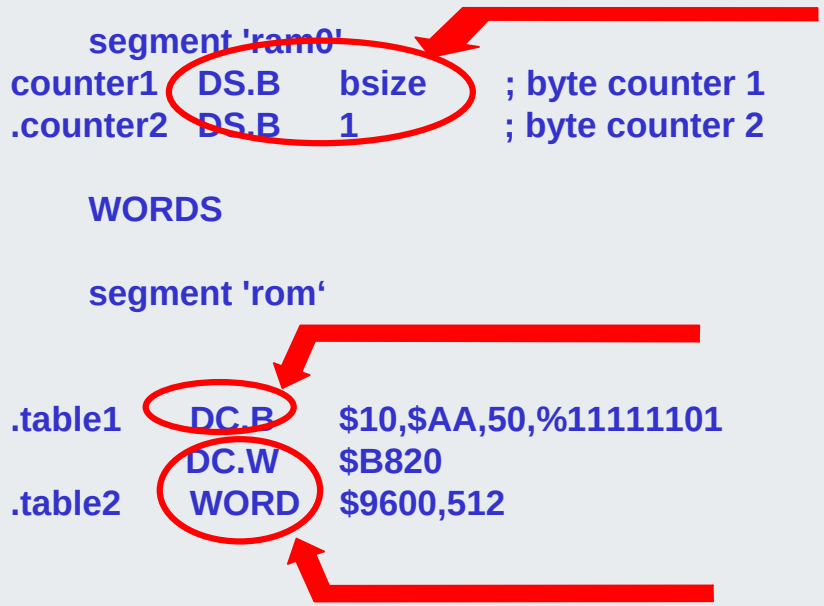
st7/

```
;*****************************************************************
;* CONST.ASM                                  date: 01/05/2000    *
;* Device : ST72254                                               *
;*****************************************************************

      TITLE    "const.asm"

      MOTOROLA                    ; format for the assembly (default)

;*****************************************************************

      BYTES

;*****************************************************************
;      Public variables, constants
;*****************************************************************
```

Same format than the other asm files (cf. module1.asm)

EXAMPLE
Const.asm (2)

```
        PUBLIC counter1
        BYTES

bsize       EQU     3           ; blocks size bsize = 3 decimal

        segment 'ram0'
counter1    DS.B    bsize       ; byte counter 1
.counter2   DS.B    1           ; byte counter 2

        WORDS

        segment 'rom'

.table1     DC.B    $10,$AA,50,%11111101
            DC.W    $B820
.table2     WORD    $9600,512


.label      EQU     {table1+2}

        end
```



Reserved 3 and 1 bytes space in ram0 counter1 & counter2 will be touched using the short addressing mode

Define byte(s) in object code

DC.W defines word(s) in code, MSB first WORD defines word(s) in code, LSB first

EXAMPLE
Memory Allocation

```
        segment 'ram0‘

counter1    DS.B        bsize           ; byte counter 1
.counter2   DS.B        1               ; byte counter 2
```

```
        WORDS

        segment 'rom‘

.table1     DC.B        $10,$AA,50,%11111101
            DC.W        $B820
.table2     WORD        $9600,512
.label      EQU         {table1+2}

        end
```

Label	Address	Content
counter1	$80	XX
	$81	XX
	$82	XX
counter2	$83	XX
…	…	…
table1	$E000	$10
	$E001	$AA
label	$E002	50
	$E003	$FD
	$E004	$B8
	$E005	$20
table2	$E006	$00
	$E007	$96
	$E008	$00
	$E009	$02

EXAMPLE
Const.inc

```
;*****************************************************************
;* CONST.INC                                      date: 01/05/2000*
;* Device : ST72254                                               *
;*****************************************************************

      TITLE    "CONST.INC"

;*****************************************************************
      EXTERN      counter1.b      ; counter1's address is a byte
      EXTERN      counter2.b      ; counter2's address is a byte
      EXTERN      table1.w        ; table1's address is a word
      EXTERN      table2.w        ; table2's address is a word
```

These labels have been defined in the zero page ie in the segment 'ram0'

Declare external labels. They will be used in other modules (all labels have been defined as public previously)

EXAMPLE
ST72254.inc

```
;*****************************************************************
;* ST72254.INC                                    date: 01/05/2000 *
;* Device : ST72254                                                *
;*****************************************************************
      TITLE    "ST72254.INC"


      EXTERN        PCDR.b              ; PCDR's address is a byte
      EXTERN        PCDDR.b             ; PCDDR's address is a byte
      EXTERN        PCOR.b              ; PCOR's address is a byte

      EXTERN        PBDR.b              ; PBDR's address is a byte
      .....
```

Same format than the other inc files (cf. const.inc)

THE ASSEMBLER: ASM
Defining macros (1)

- A macro is a faster way to execute code: there is no stacking for return addresses (less stack activity)

- But the program size is bigger using macros than subroutines:
  - **Each time you invoke a macro to do a particular job, the whole macro assembly code is inserted into your source code**
  - **Trade off between code size and execution time**

- Very usefull when a small size of code must be used repeatedely

THE ASM ASSEMBLER
Defining macros (2)

- macro_name MACRO [parameter_1] [, parameter_2...]
  [LOCALlabel_name]
  [Body_of_the_macro]
  MEND
- Example:

```
get_io      MACRO
      LOCAL loop
loop  ld    a, $C000    ;if $C000 content not equal to 0 then...
      jrne  loop        ; jp loop. If equal, end macro
      MEND
```

The "LOCAL" directive avoids to have labels inside the macro duplicated (because the same macro can be called several time in the same module)

THE ASSEMBLER: ASM
Conditional assembly

- Conditional assembly is used to ignore or select whole area of code
- Example:

```
#IFDEF HIGH
        #IF {HIGH eq 1}
                ld      A, #const.H
                #ELSE
                ld      A, #const.L
                 #ENDIF
#ENDIF
```

HIGH can be defined either:

- **In the code (#define directive)**
- **in the command line using the -D switch (-D HIGH 1)**

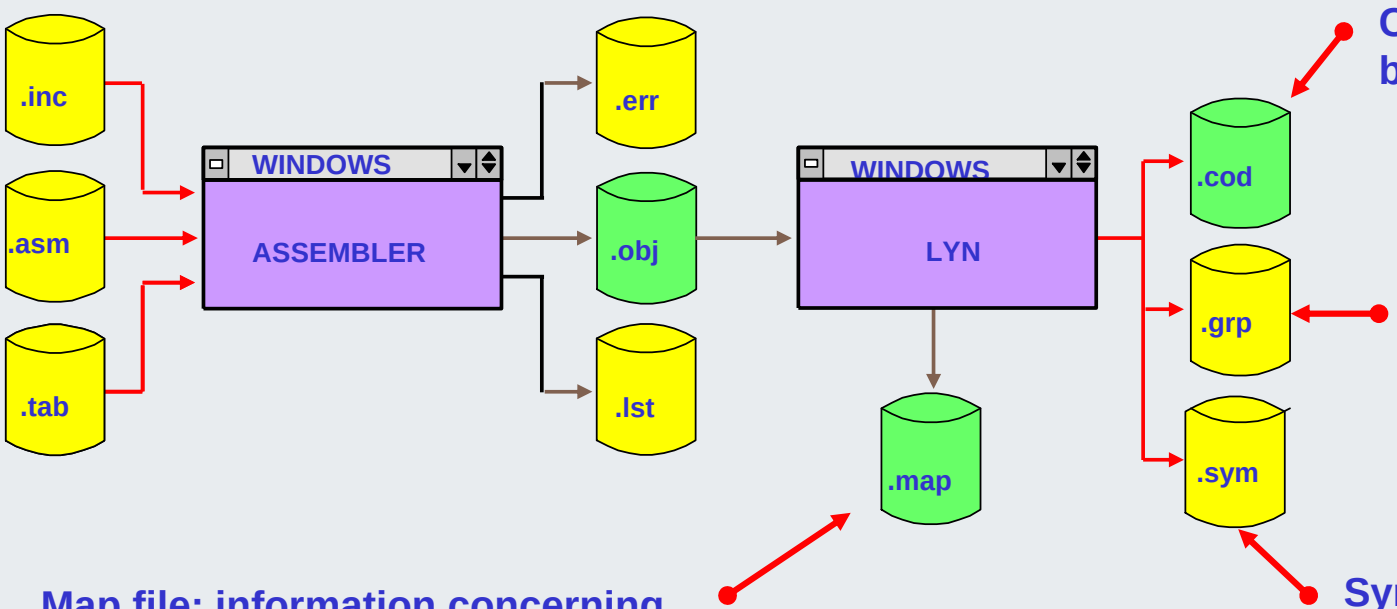
THE LINKER: LYN
Overview (1)

- The linker:
  - **Concatenates all the objects together to create a .cod file**
  - **Checks and resolves all the PUBLIC & EXTERNAL references**
  - **Checks and places the segments**
  - **Scans the libraries**

- Example:

```
LYN st72254+module1+module2+const, appli;
or
LYN @list.rsp
where list.rsp can be:
st72254+module1+module2+const,appli;
```

THE LINKER: LYN
Overview (2)



Cod file: to be used by the formatter

Grp file: not used by ST tool chain

Map file: information concerning segments, labels and classes

Sym file contains all the public symbols (symbol table list)

LYN <.obj file> [+<.obj file>...], [<.cod file>], [<lib>], [+<lib>];

THE FORMATTER: OBS
Overview

- Obsend is a general file formatter which converts the .cod file into an executable format useable by the programmers and debuggers

- Example:

```
OBSEND appli,f,appli.s19, s
```

  - ✓It will generate file appli.s19 containing the code from appli.cod in "Motorola S-record" format

THE FORMATTER: OBS Option...

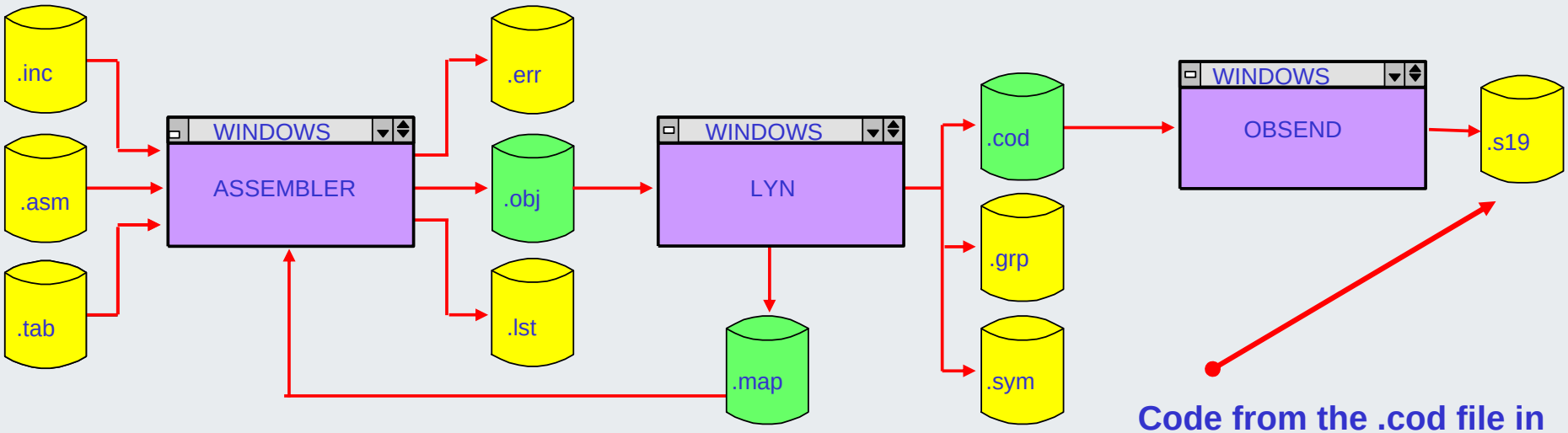
OBSEND <file>, [destination type] [,<args>], <format>

f: file
v: video

Name of the output file

- Format: specify the output format
  - ✓ i — Intel Hex
  - ✓ i32 — Intel Hex with 32-b data per line
  - ✓ ix — Intel Hex extended
  - ✓ s — Motorola S-record (1 byte/address)
  - ✓ x — Motorola S-record extended
  - ✓ 2 — ST S-record 2
  - ✓ 4 — ST S-record 4
  - ✓ f — Filled straight binary format
  - ✓ g — GP industrial binary format

THE FORMATTER: OBS
Overview



Code from the .cod file in Motorola format (in this example)

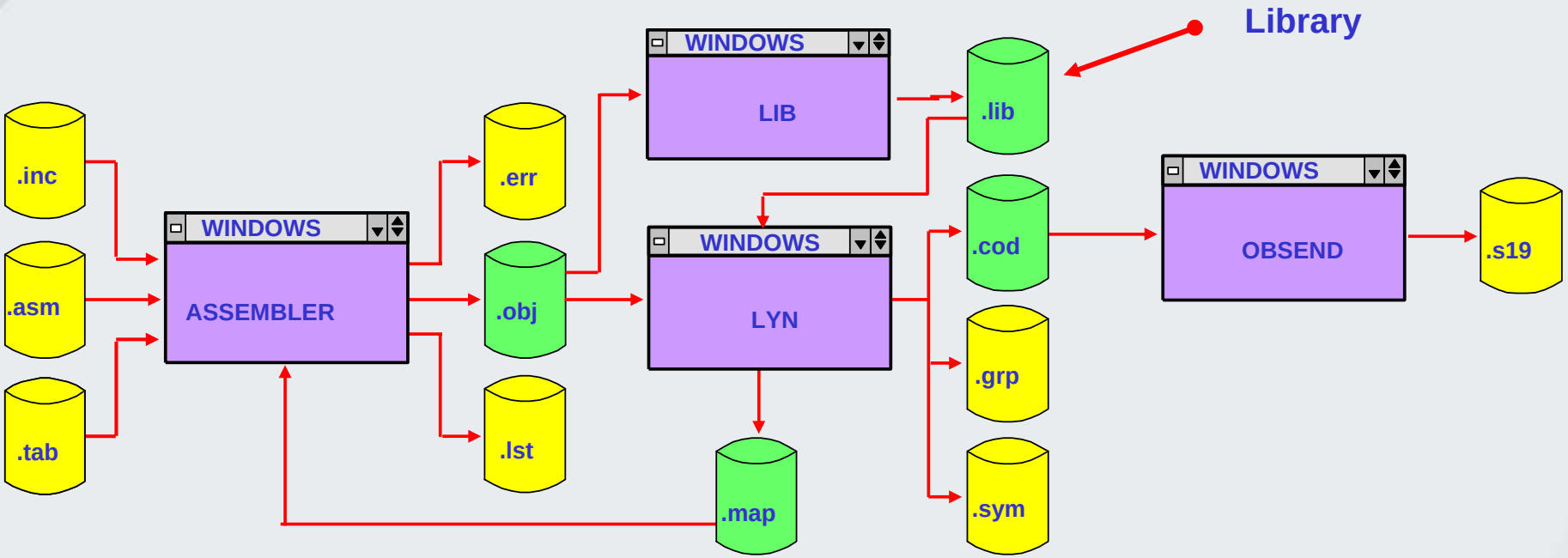
OBSEND <file>, [destination type] [,<args>], <format>

THE LIBRARIAN: LIB
Overview (1)

- Combines small but useful object files (created by the user) in a library

  - **Options:**
    - ✓ **+filename** **adds an object module to the library**
    - ✓ **-filename** **deletes an object module from the library**
    - ✓ **!filename** **updates an object module in the library**
    - ✓ ***filename** **copies an object module to a file**
    - ✓ **?** **lists the library content**
    - ✓ **X** **exits to dos prompt**

THE LIBRARIAN: LIB
Overview (2)



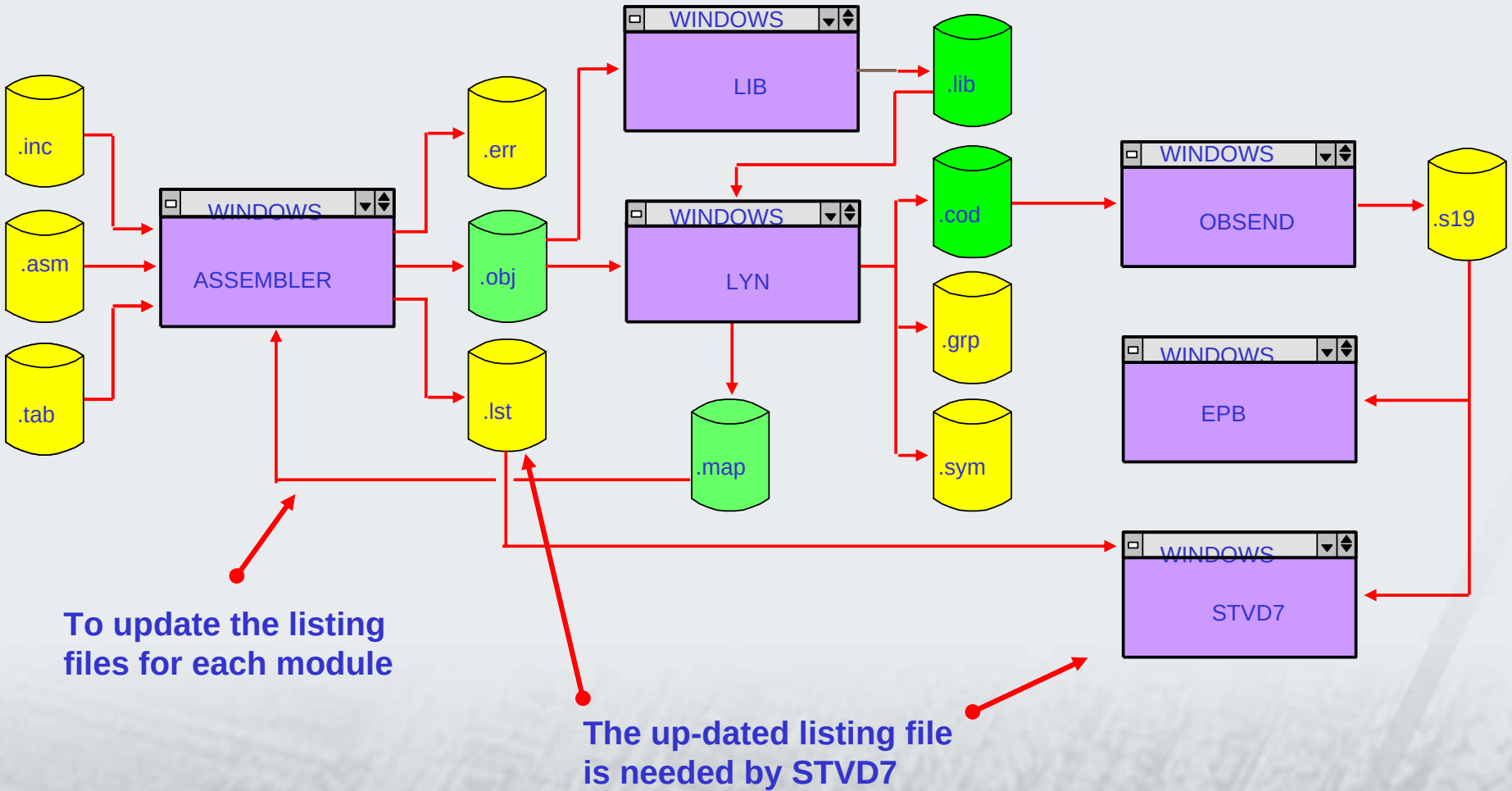
LIB <library_name>

PROGRAMMING TIPS
Batch file (1)

- As the current assembler is Crash-Barrier, all the modules are assembled separately. That means that in order to get fixed references (and then no more relative addresses), modules have to be assembled a second time after the link

- "-FI" option enables to perform an absolute patch on the desired listing (.lst, .cod and .map): updates the symbol table in the listing file

PROGRAMMING TIPS
Batch file (2)



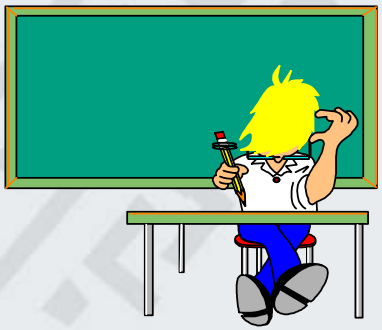
PROGRAMMING TIPS
Batch file (3)

- Example:

To have the correct result message (Build succeeded or failed) in the Output window

```
DEL *.s19
DEL *.obj
ASM -li st72254.asm
ASM -li module1.asm
ASM -li module2.asm
ASM -li const.asm
LYN st72254.obj+module1.obj+module2.obj+const.obj,appli.cod;
ASM st72254 -sym -fi=appli.map
ASM module1 -sym -fi=appli.map
ASM module2 -sym -fi=appli.map
ASM const -sym -fi=appli.map
OBSEND appli.cod,f,appli.s19, s
```

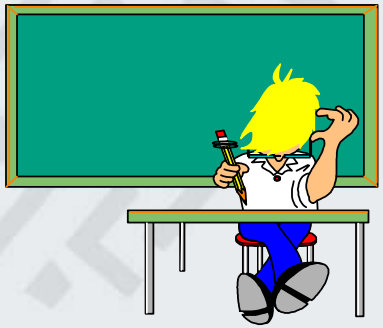
This time, the assembler does not create a .obj file but uses the mapfile created by the linker to fix all the addresses



Exercise in assembly langage (1)

Constant.asm

```
                WORDS
                segment 'rom‘

                        PUBLIC label
baud_rate               WORD        9600, 18200, main
.romtable               DC.B        125, $A8, %01110100, "StRiNg", '/'
.constant               DC.W        568, $FF86
label                   EQU         {romtable+2}

                        BYTES
                      segment 'ram0‘

.ramtable               DS.B        8
.variable               DS.W        2
.count                  DS.B        1
```

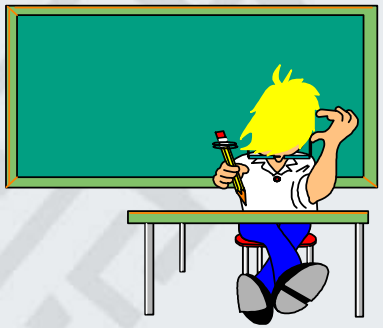


Exercise in assembly langage (2)

Module1.asm

```
            WORDS
        segment 'rom'
.main
                    ld      A, ramtable
                    ld      A, romtable
                    jra     main
                    end
```

Tips: 9600d = 2580h; 18200d = 4718h
"StRiNg" = 53h, 74h, 52h, 69h, 4Eh, 67h
'/' = 2Fh
568d = 238h
1Bh+F9h = 14h; +:addition on 8 bit words



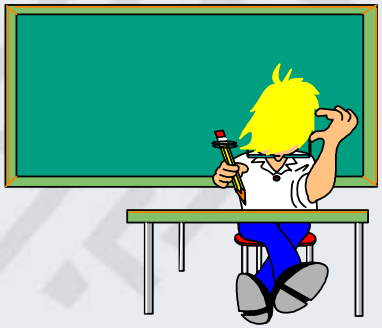
Exercise in assembly langage (3)

- Fill the memory map according to the files'constant.asm' &'module1.asm' shown in the previous slides.

N.B. 'Constant.obj' is linked first.

Label	Address	Content
	$80	
	$E000	

Label	Address	Content



Exercise in assembly langage (4)

Fill the memory map according to the files'constant.asm' &'module1.asm' shown in the previous slides.

N.B. 'Constant.obj' is linked first.

Label	Address	Content
ramtable	$80	XX
variable	$88	XX
count	$8C	XX
	$E000	80
		25
		18
		47
		14
		E0
romtable	$E006	7D
		A8
label	$E008	74
		53
		74
		52

Label	Address	Content
		69
		4E
		67
		2 F
constant	$E010	O2
		38
		FF
		86
main	$E014	B6
		80
	$E016	C6
		E0
		O6
	$E019	20
	$E01A	F9
	$E01B	