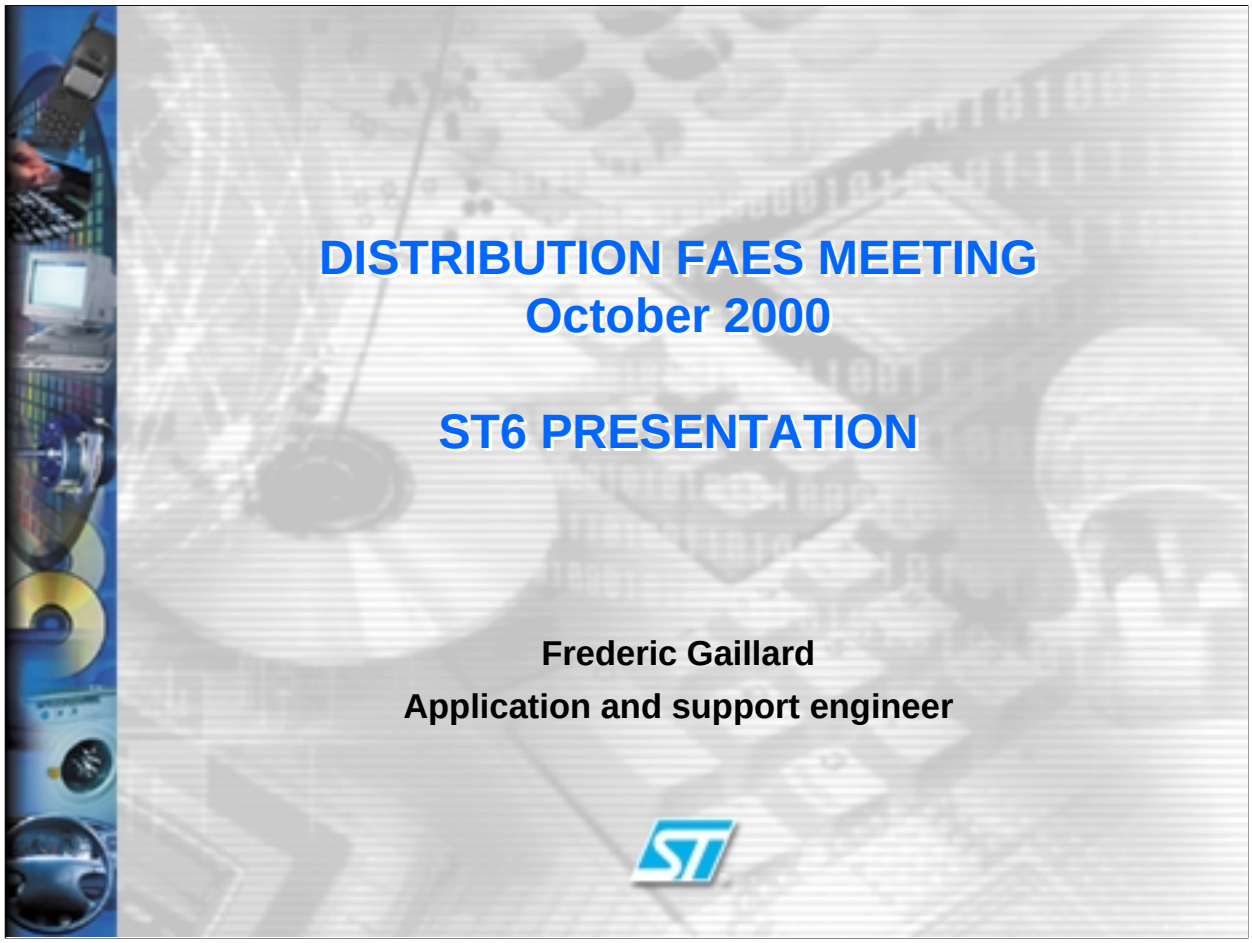
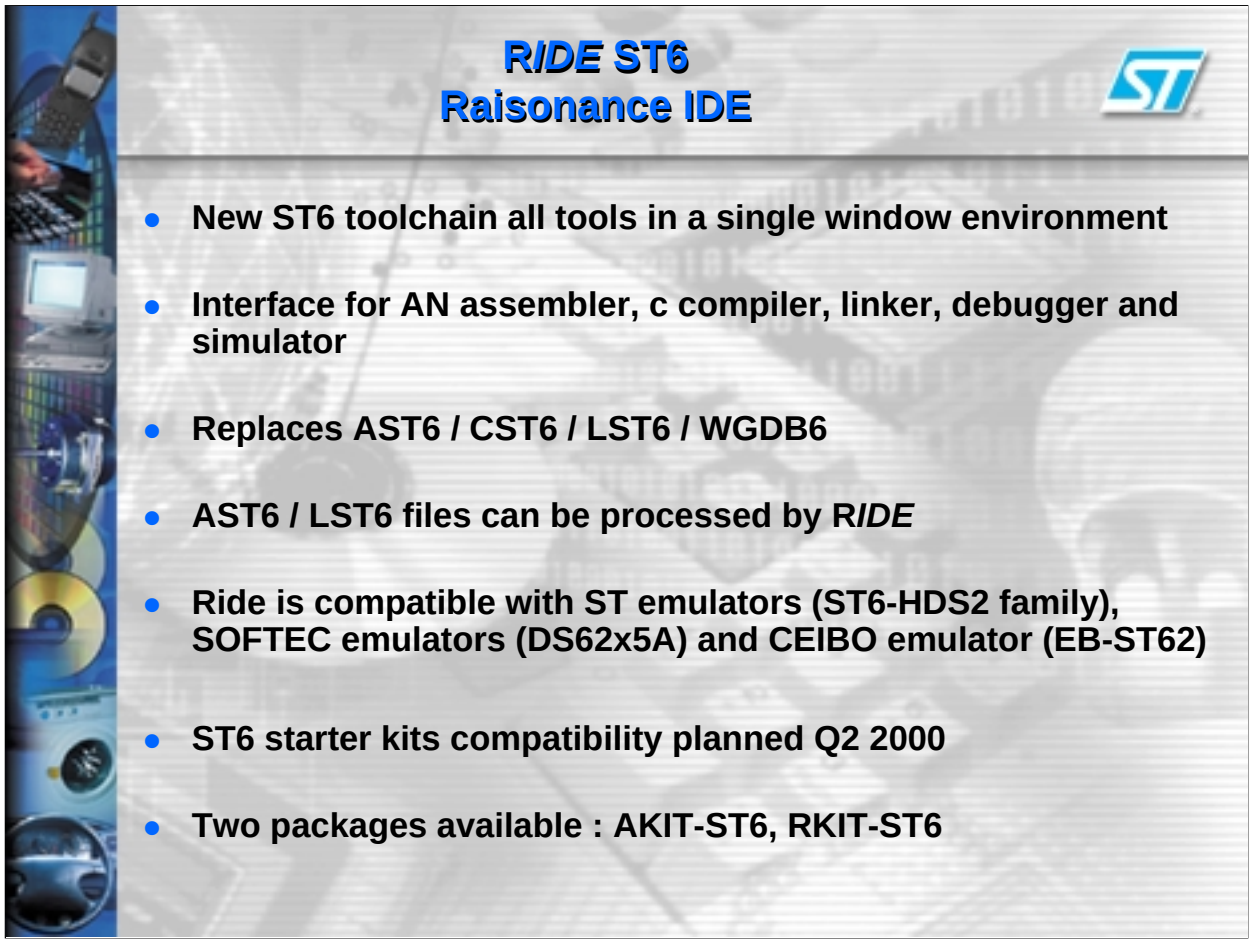

DISTRIBUTION FAES MEETING
October 2000
ST6 PRESENTATION
Frederic Gaillard
Application and support engineer

ST62 DEVELOPMENT TOOLS

1. R*IDE*
2. MAST6 ASSEMBLER
3. From AST6/LST6 to R*IDE*
4. RCST6 C COMPILER
5. CEIBO and SOFTEC emulators

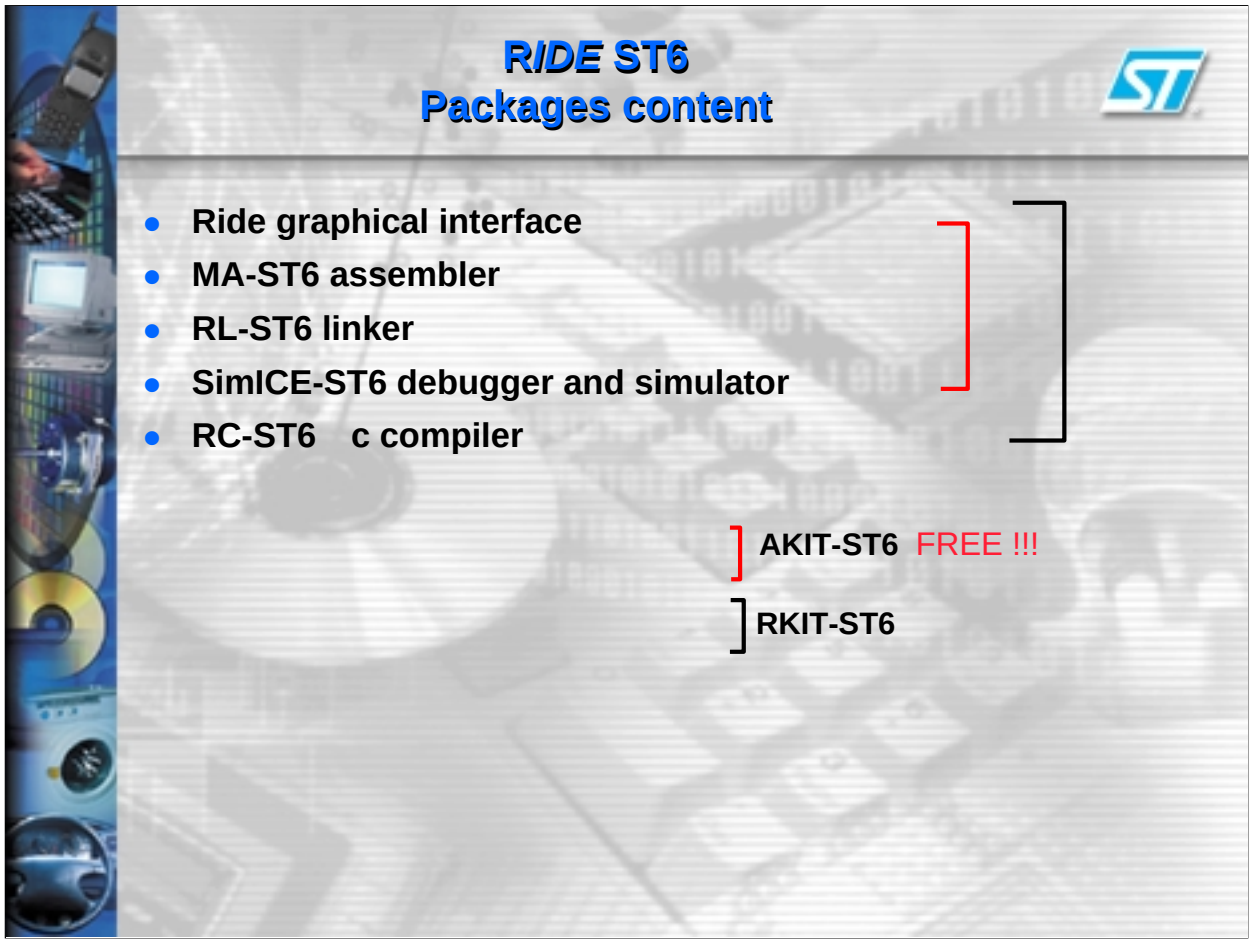

RIDE ST6
Raisonance IDE
New ST6 toolchain all tools in a single window environment
Interface for AN assembler, c compiler, linker, debugger and simulator
Replaces AST6 / CST6 / LST6 / WGDB6
AST6 / LST6 files can be processed by RIDE
Ride is compatible with ST emulators (ST6-HDS2 family), SOFTEC emulators (DS62x5A) and CEIBO emulator (EB-ST62)
ST6 starter kits compatibility planned Q2 2000
Two packages available : AKIT-ST6, RKIT-ST6

RIDE ST6
Raisonance IDE

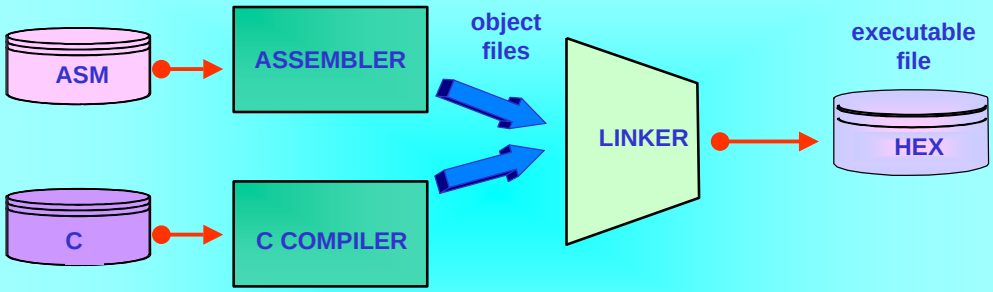
- **Text editor**
  - Syntax highlighting
  - GREP, INDENT,FIND, 'Search for matching delimiter' functions
  - Available in debug session
- **Project manager**
  - MAKE, BUILD
  - Tree-structured application
- **Tools integrator**
  - Predefined tools
  - Custom tools

RIDE ST6
Raisonance IDE

- **DEBUGGER INTERFACE**
  - **Numerous views available :**
    - **Source code**
    - **Disassembly code**
    - **Symbols**
    - **Peripherals**
    - **....**
  - **Numerous debug functions :**
    - **Breakpoints**
    - **Trace**
    - **Stimulus for the simulator**
    - **.....**


RIDE ST6
Packages content
Ride graphical interface
MA-ST6 assembler
RL-ST6 linker
SimICE-ST6 debugger and simulator
RC-ST6 c compiler
AKIT-ST6 FREE !!!
RKIT-ST6

RIDE ST6
Programming tools



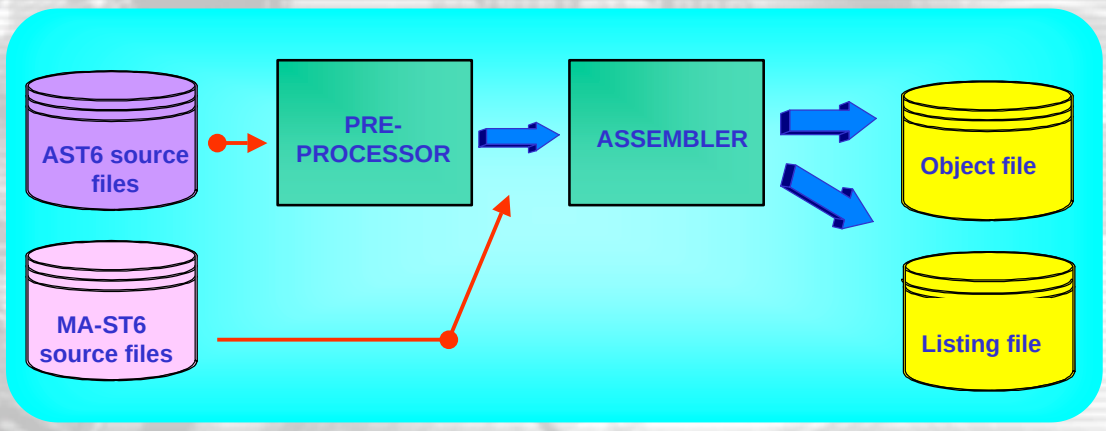
FULLY INTEGRATED INTO RIDE

ST62 DEVELOPMENT TOOLS

1. R*IDE*
2. MAST6 ASSEMBLER
3. From AST6/LST6 to R*IDE*
4. RCST6 C COMPILER
5. CEIBO and SOFTEC emulators

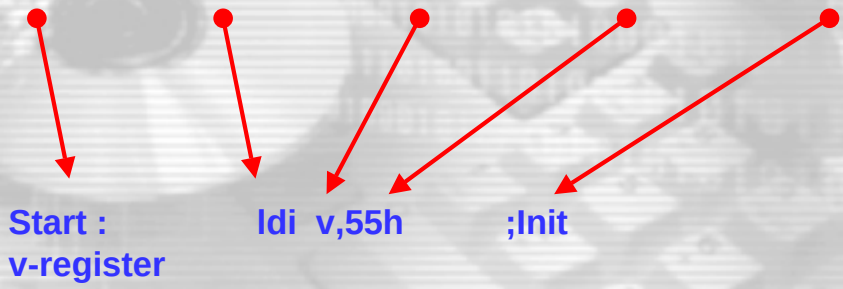
RIDE ST6
MA-ST6 assembler

- Translates ST6 assembly mnemonics into machine code
- Source compatible with STMicroelectronics AST6/LST6
- Generates a listing file and a relocatable object file



RIDE ST6
MA-ST6 assembler

- ASCII File with extension ".ST6"
- Each line has up to 4 fields:
- [LABEL] OPERATION [OPERAND,[OPERAND]] ;[COMMENT]



MA-ST6 ASSEMBLER
Numbers and character constants

- Default base for numbers is decimal
- Base can be overrided by adding a suffix:
  - "b" or "B" -> binary (0,1)
  - "o" or "O" -> octal (0-7)
  - "h" or "H" -> hexadecimal (0-9,A-F or a-f)
- Hexadecimal numbers must start with a 0 to avoid confusion with symbol names
- A character constant is an ascii character enclosed in single quotes
- Its value is the 8-bit ASCII code of the character
  Example: 'A' = 65 = 41h

MA-ST6 ASSEMBLER
Directives

- Used to control the way the assembler will process instructions
- Several categories of directives:
  - Symbol definition in data space (DATA)
  - Symbol definition in program space (DB, DW, ASCII ...)
  - Symbol assignment (EQU, SET)
  - Hardware-related directives (SEGMENT, $DATAPAGING, $PROGPAGING,...)
  - Linking directives (GLOBAL, EXTERN...)
- NOTES: They can be written in uppercase or lowercase

MA-ST6 ASSEMBLER
Segment

- **DEFINITION**
  - **In order to optimize the code size and the RAM used, MAST6 includes a directive that enables to define part of the memory (code or data) as segments.**
  - **An ABSOLUTE segment has a precise address.**
  - **A RELOCATABLE segment has no precise address and it will be placed physically by the linker.**

MA-ST6 ASSEMBLER
Relocatable segment

- **SEGMENT DECLARATION**

  Syntax: *Seg_name* SEGMENT *Seg_type* [PAGE *num*]

  - *Seg_type* possible values:
    - **CODE** — **segment located in program space**
    - **DATA** — **segment located in RAM space**
    - **EEPROM** — **segment located in EEPROM space**
    - **DATAROM** — **segment located in program space in a 64-byte window**

MA-ST6 ASSEMBLER
Relocatable segment

- PAGE *num* possible values:
  - **CODE and DATAROM (program space) : *0, static, 2, 3, auto***
  - **DATA (RAM space): *static, 1, 2, auto***
  - **EEPROM (EEPROM space) *: 1,2, auto***

- **SEGMENT SELECTION**

  Syntax: RSEG *Seg_name*

The current segment remains selected until a new one is found

MA-ST6 ASSEMBLER
Relocatable segment

- **EXAMPLE**

```
Codseg      SEGMENT     CODE        ; relocatable code segment
data1seg    SEGMENT     DATA        ; relocatable data segment
data2seg    SEGMENT     DATA        ; relocatable data segment

RSEG        data1seg                ; data1seg segment selection
counter:    DSB         1           ; reserve 1 byte in data1seg

RSEG        codseg                  ; codseg segment selection
ldi         A, 55h
ld          A, X
nop
```

MA-ST6 ASSEMBLER
Absolute segment (1/2)

- **CODE SEGMENT DECLARATION (1/2)**

  Syntax : CSEG AT *address* [PAGE *num*]
  PAGE has to be specified if the address is in the banking area

- **EXAMPLE**

```
Codseg    SEGMENT  CODE   ;relocatable segment declaration

CSEG      AT 880h          ;Absolute code segment declaration
jp Label

RSEG      codseg           ;codseg segment selection
nop

CSEG                       ;address in now 882h
```

MA-ST6 ASSEMBLER
Absolute segment

- **CODE SEGMENT DECLARATION (2/2)**
  Syntax: *Seg_name* SEGMENT *Seg_type* AT *address*

- **CODE SEGMENT SELECTION**
  Syntax: ASEG *Seg_name*

  ASEG can be used only if the segment was declared with AT

MA-ST6 ASSEMBLER
Absolute segment

- **DATA SEGMENT DECLARATION**

  Syntax: DSEG AT *address* [PAGE *num*]
  PAGE has to be specified if the address is in the banking area

- **EXAMPLE**

```
DSEG    AT      90h     ; Data segment starts at 90h
```

- **EEPROM SEGMENT DECLARATION**

  Syntax: ESEG AT *address* PAGE *num*

  The use of PAGE is mandatory

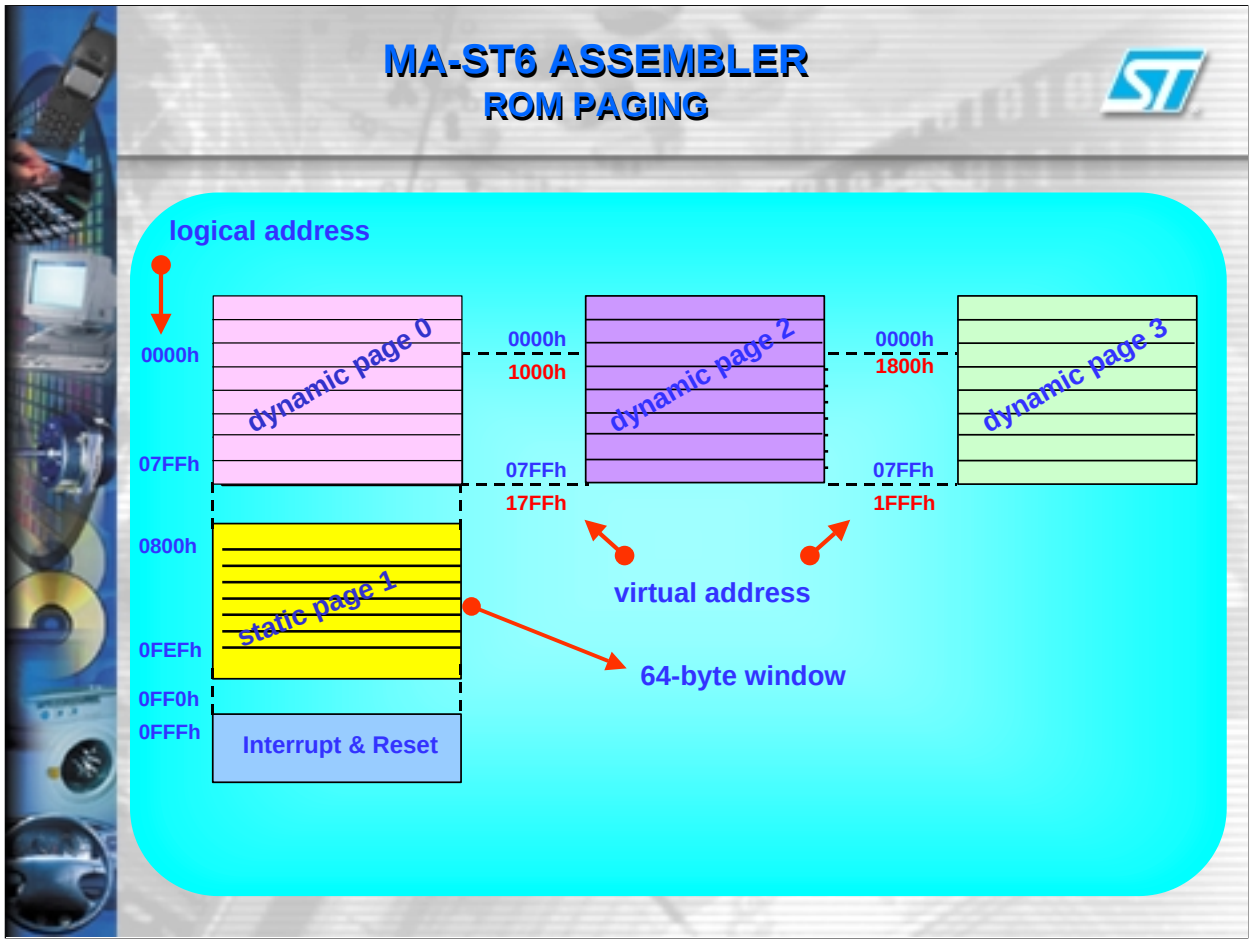
MA-ST6 ASSEMBLER
Memory management

- How To Enable Rom And Ram Banking?
- How To Use Rom Windowing ?
- How To Reserve Memory Spaces ?
- How To Initialize Memory Spaces In Program Space ?
- How To Assign Symbols To Addresses, Registers Or Values ?

MA-ST6 ASSEMBLER
ROM PAGING

- $PROGPAGING
  - Enables the use of Bank switching mode on the code space
- $NOPROGPAGING (DEFAULT)
  - Disables Bank switching mode
- #PAGE (*expr*)
  - Enables to get the ROM page number where *expr* is defined

MA-ST6 ASSEMBLER
ROM PAGING



MA-ST6 ASSEMBLER
ROM WINDOWING

- No directive needed to enable Rom windowing mecanism

- #WINDOW (*expr*)
  Used to set DWR value
  Determines the 64-byte window number
  where expr is located

- #WINOFFSET(*expr*)
  Returns the relative address of expr in the window

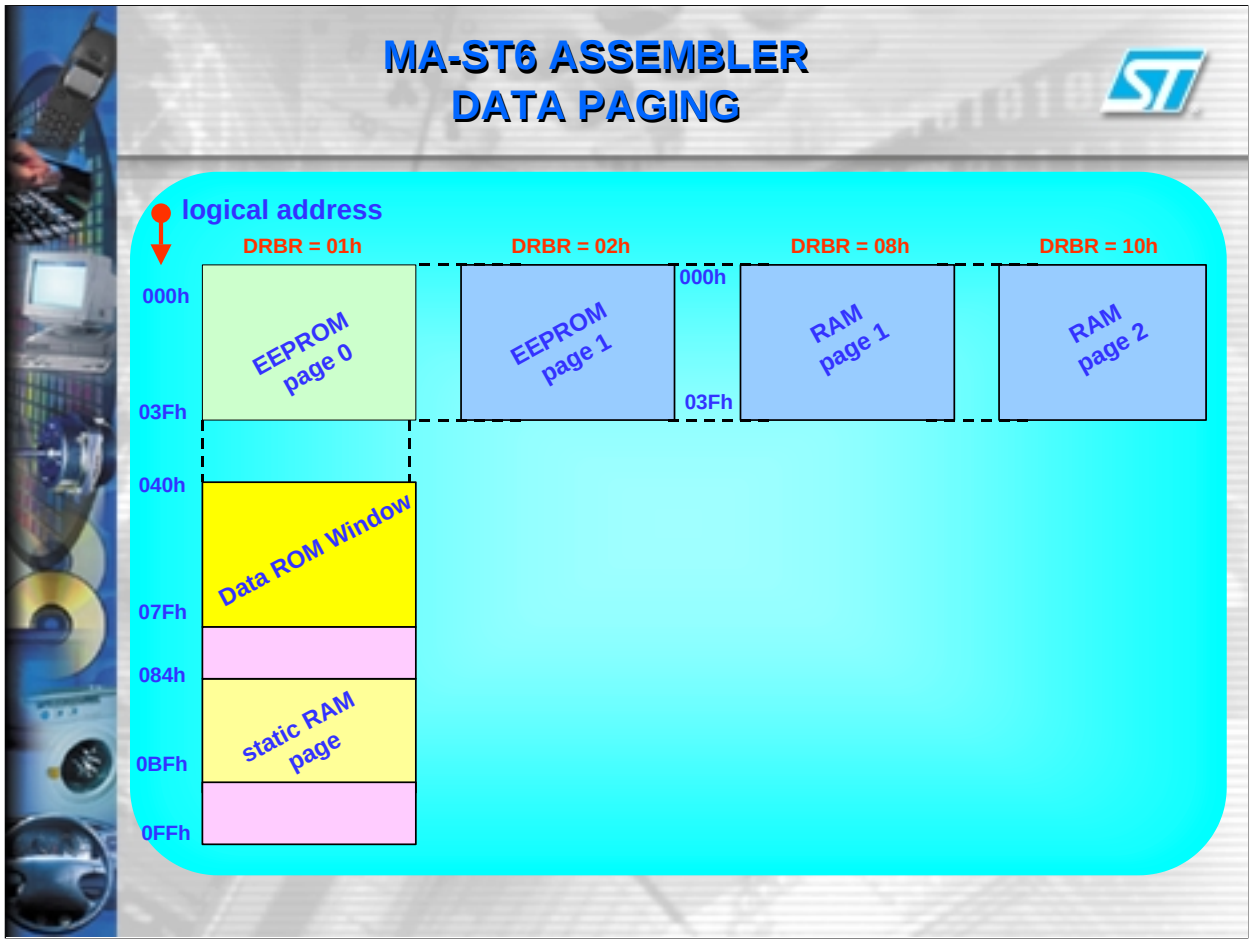
ROM PAGING & WINDOWING
EXAMPLE

```
                $PROGPAGING      ; Enable bank switching for ROM

codseg0         SEGMENT  CODE PAGE 0        ; codseg0 will be mapped in
                                              page 0
codseg1         SEGMENT  CODE PAGE static   ; codseg1 will be mapped in
                                              page 1
Table           SEGMENT CODE DATAROM        ; Table will be located in a 64-
                                              bytes window

RSEG            Table                       ; Select Table
tab:            DB    0Dh, 0Eh, 0Fh

RSEG            codseg0                     ; Select codseg0
SUBR1:          ldi DWR, #window(tab)       ; Set ROM Window Register
                ldi X, #winoffset(tab)      ; X= address of first element in
                                              the table
                ld A,(X)                    ; A= 0Dh
                ret
RSEG            codseg1                     ; Select codseg1
                ldi PRPR, #page(SUBR1)      ; Set PRPR value where SUBR1
                                              is located
                call SUBR1
        ......
```

MA-ST6 ASSEMBLER
DATA PAGING

- $DATAPAGING
- $NODATAPAGING (DEFAULT)
- #PAGE (*expr*)
  - Used to set the DRBR value enables to get the ram or EEPROM page number where *expr* is defined

MA-ST6 ASSEMBLER
DATA PAGING



MA-ST6 ASSEMBLER
DATA PAGING

- **EXAMPLE**

```
                $DATAPAGING          ; Enable bank switching for RAM

codseg          SEGMENT  CODE
dataseg         SEGMENT  DATA

RSEG     dataseg                     ; Select dataseg
reg1:  DS    1                         ; Reserve 1 byte in dataseg
reg2:  DS    1                         ; Reserve 1 byte in dataseg

RSEG     codseg                      ; Select codseg
                ldi DRBR, #page (reg1)  ; Set DRBR value where reg1 is located
                ldi reg1, 55h
                Inc reg2
```

MA-ST6 ASSEMBLER
Symbol definition

- **RAM SYMBOL DEFINITION**
  Syntax : *Symb_name* DATA *address*

  - **Associates a RAM address to a symbol, no physical location is reserved**

  - **If *address* is in the range 00-3Fh, the definition must be precedeed by directives:**

    $DATAPAGING then
    $DATAPAGENUMBER(*val*) where *val* = 1 or 2 to select the proper dynamic page (refer to datasheets)

    $DATAPAGENUMBER is only used when variables are defined using DATA

MA-ST6 ASSEMBLER
Symbol definition

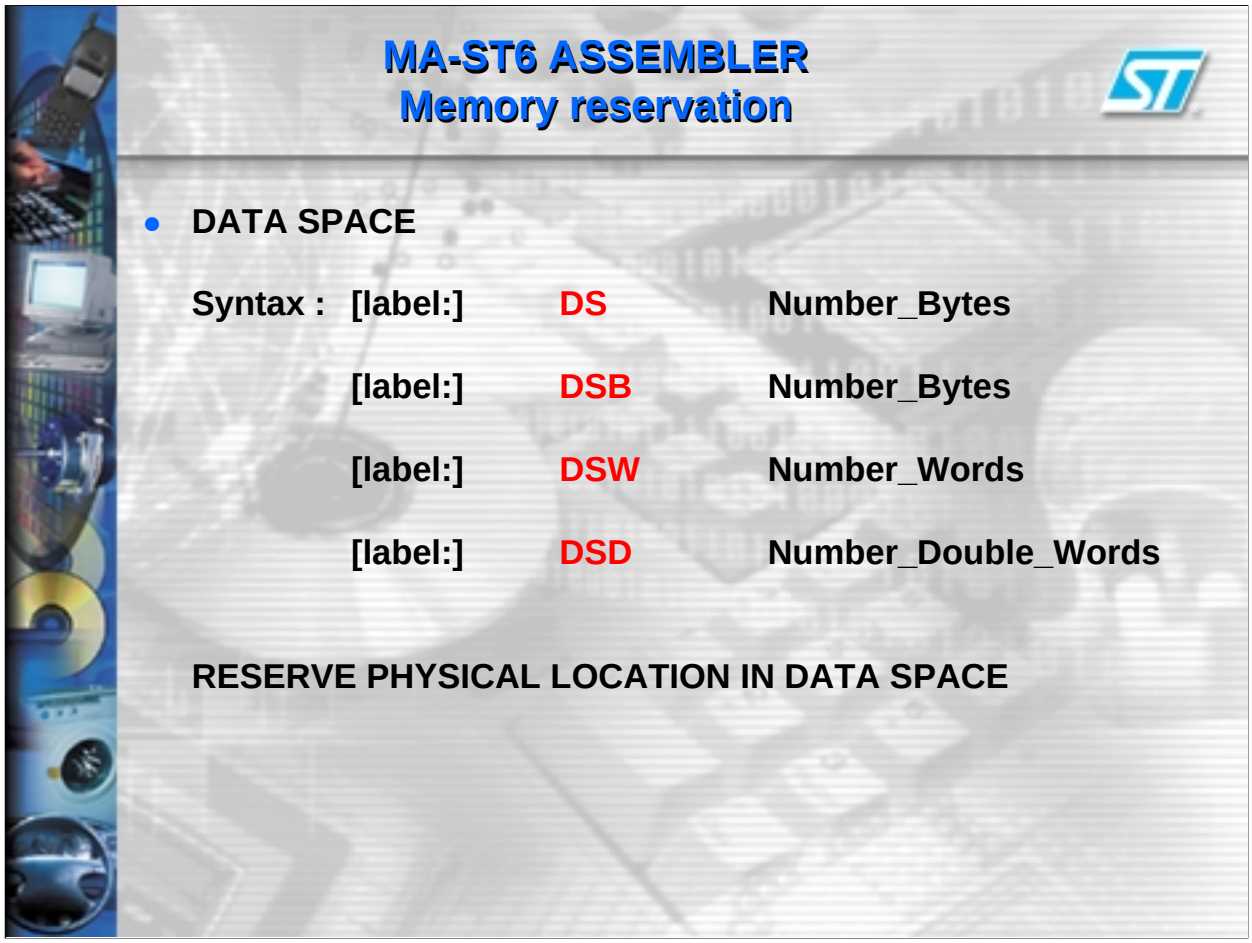
- **EEPROM SYMBOL DEFINITION**
  Syntax : *Symb_name* EEPROM *address*
  - **Associates an EEPROM *address* to a symbol no physical location is reserved**
  - ***Address* is in the range 00-3Fh**
  - **The definition must be precedeed by directive:**

    $EEPROMPAGENUMBER(*val*) where *val* = 0 or 1 to select the proper dynamic page (refer to datasheets)

MA-ST6 ASSEMBLER
Symbol definition

- **EXAMPLE**

```
X           DATA        80h
DWR         DATA        0C9h

$DATAPAGENUMBER(1)      ; Following symbols are in RAM page 1
                        ; DRBR=08h
Var1        DATA        10h
Var2        DATA        11h

$EEPROMPAGENUMBER(0); Following symbols are in EEPROM page 0
                        ; DRBR=01h
eepresult   DATA        05h
```


MA-ST6 ASSEMBLER
Memory reservation
DATA SPACE
Syntax : [label:] DS Number_Bytes
[label:] DSB Number_Bytes
[label:] DSW Number_Words
[label:] DSD Number_Double_Words
RESERVE PHYSICAL LOCATION IN DATA SPACE

MA-ST6 ASSEMBLER
Memory reservation

- **EXAMPLE**

```
DSEG      AT    90h

Table1:   DS    6       ; Reserve 6 bytes for Table1 90h - 95h
Table2:   DSB   6       ; Reserve 6 bytes for Table1 96h - 9Bh
Buffer1:  DSW   2       ; Reserve 4 bytes for Table1 9Ch - 9Fh
Buffer2:  DSD   1       ; Reserve 4 bytes for Table1 0A0h - 0A3h

CSEG      AT    800h
LDI       A, Table1     ; Table 1 initialization
LD        X, A
LDI       A, 0FFh
LD        (X), A
```

MA-ST6 ASSEMBLER
Memory initialization

- **Constant definition in program space**

  - **BYTES**
    Syntax:[label:] DB *exp [,exp]* **where *exp* is a 8-bit value**

    - **EXAMPLE**

```
CSEG        AT  800h

Table1:     DB      0, 1, 2, 3, 'Raisonance'
```

MA-ST6 ASSEMBLER
Memory initialization

- **WORD**
  Syntax: [label:] DW *exp [,exp]* **where exp is a 16-bit value**

  - **EXAMPLE**

```
CSEG      AT 830h

Table20:  24h, 25h      ; Memory content      30h -> 00
                        ;                     31h -> 24
                        ;                     32h -> 00
                        ;                     33h -> 25
```

MA-ST6 ASSEMBLER
Memory initialization

- **CHARACTER OR STRING**
  Syntax: [label:] DB *'exp'* [*,'exp'*]

- **DEFINE AN ASCII CHARACTER OR A STRING**
  If double quotes are used, it defines a null terminated string

- **Message1: DB 'OK‘ = Message1 ASCII "OK"**
- **Message2: DB "YES" = Message2 ASCIIZ"YES"**

MA-ST6 ASSEMBLER
Symbol assignment

- **Directives enable to assign a symbol to a numerical value, register name or register bit**
- **EQU**
  Syntax: *Symb* EQU *exp*
  - **Symbol can not be redefined nor changed it must be unique**
  - **EXAMPLE**

```
Timer_tick    EQU    5
Var           EQU    3
Pointer       EQU    X
Const         EQU    Timer_tick*Var
```

MA-ST6 ASSEMBLER
Symbol assignment

- **SET**
  Syntax: *Symb* SET *exp*

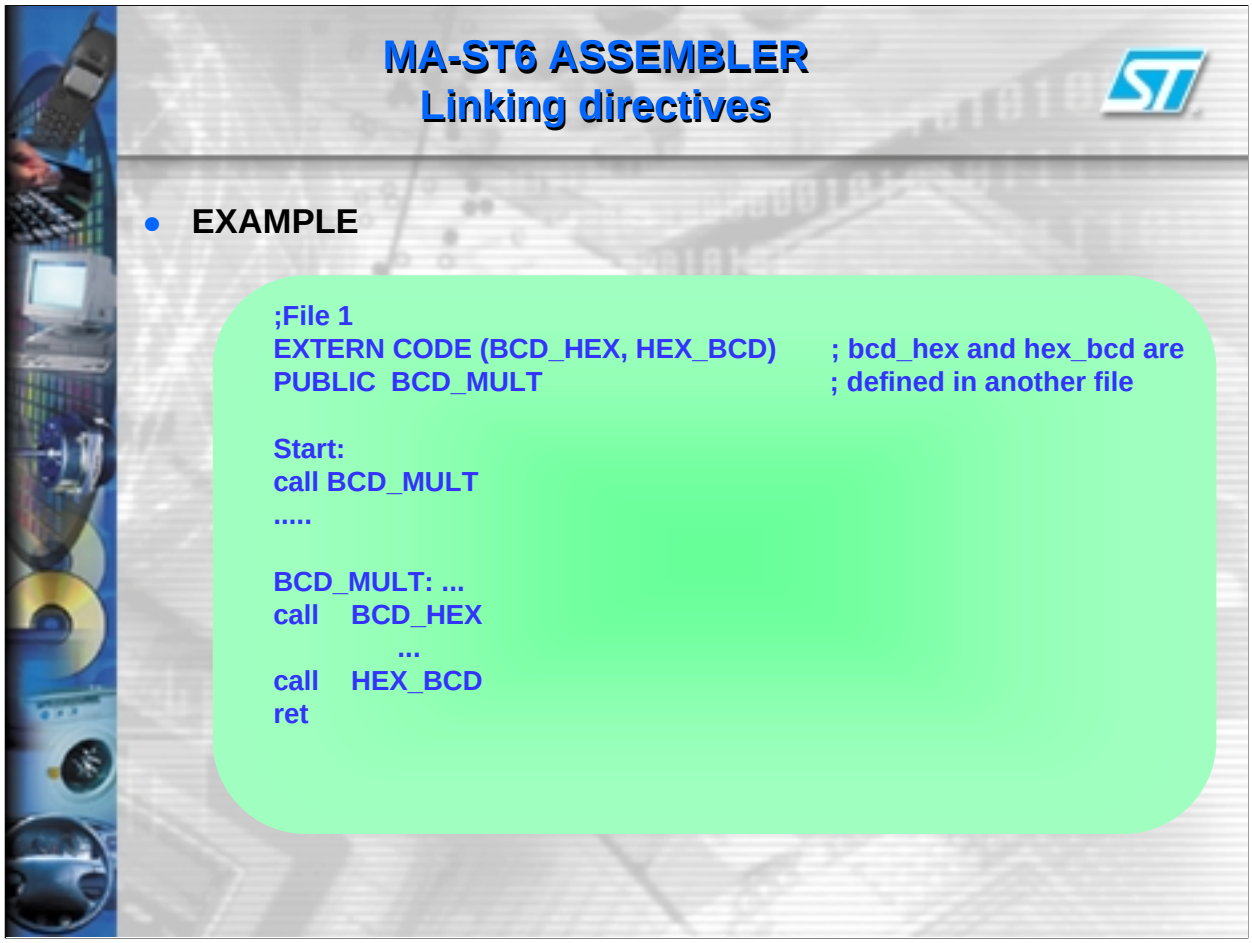
  - **Symbol can be redefined by another set statement**

  - **EXAMPLE**

```
TMZ     SET     7
Count   SET     3
Tim     SET     X
Count   SET     Count + 6
```

MA-ST6 ASSEMBLER
Linking directives

- **PUBLIC**
  Syntax: PUBLIC *symb [,symb]*
  - **Allow use of the symbol in other files symbol following public must be declared in the current module**
- **EXTERN**
  Syntax: EXTERN *Seg_type (symb [, symb...])*
  - **Used to access symbols declared in other files**
  - ***Seg_type* can be : CODE, DATA and NUMBER**


MA-ST6 ASSEMBLER
Linking directives
EXAMPLE
;File 1
EXTERN CODE (BCD_HEX, HEX_BCD) ; bcd_hex and hex_bcd are
PUBLIC BCD_MULT ; defined in another file
Start:
call BCD_MULT
.....
BCD_MULT: ...
call BCD_HEX
...
call HEX_BCD
ret

MA-ST6 ASSEMBLER
Other directives

- **INCLUDE directive**

  Syntax : $INCLUDE *(FileName)*

  - **The source of the specified file will be inserted**
  - **The file must be in the current directory or the path must be specified**

- **ORG directive**

  Syntax : ORG *expr*

  - **Specify an offset for a CODE or DATA segment**

MA-ST6 ASSEMBLER
Conditional assembly

- Code is assembled only under certain conditions
  - Useful for debugging purposes
  - Enhances macro programming

```
.IF
<expression>
……..
……..
.ELSE
……..
……..
.ENDIF
```

exp1 GTE exp2	true if exp1>= exp2
exp1 GT exp2	true if exp1> exp2
exp1 LTE exp2	true if exp1 <= exp2
exp1 LT exp2	true if exp1 < exp2
exp1 EQ exp2	true if exp1 = exp2
exp1 NE exp2	true if exp1 != exp2

ST62 DEVELOPMENT TOOLS

1. RIDE
2. MAST6 ASSEMBLER
3. From AST6/LST6 to RIDE
4. RCST6 C COMPILER
5. CEIBO and SOFTEC emulators

From AST6/LST6 to RIDE

- **ST6 MCUs ≤ 4K ROM and 128 bytes RAM**

AST6/ LST6	MAST6
. DEF	DATA
.ORG	CSEG AT
.EQU/.SET	EQU/SET
.INPUT	$INCLUDE

- **REMINDER**
  - Ride is able to process AST6/LST6 files
  - All ST6 peripheral control registers are automatically defined in Ride
  - No batch file needed using Ride

From AST6/LST6 to RIDE

- **ST6 MCUs > 4K ROM and 128 bytes RAM**

AST6/ LST6	MAST6
. PP_ON, label.p	$PROGPAGING, #PAGE(label)
.DP_ON, label.p	$DATAPAGING, #PAGE(label)
.W_ON, label.w, label.d	#WINDOW(label), #WINOFFSET(label)
.SECTION n	Relocatable segment declaration and selection
.Window, .Windowend	Relocatable segment with DATAROM type

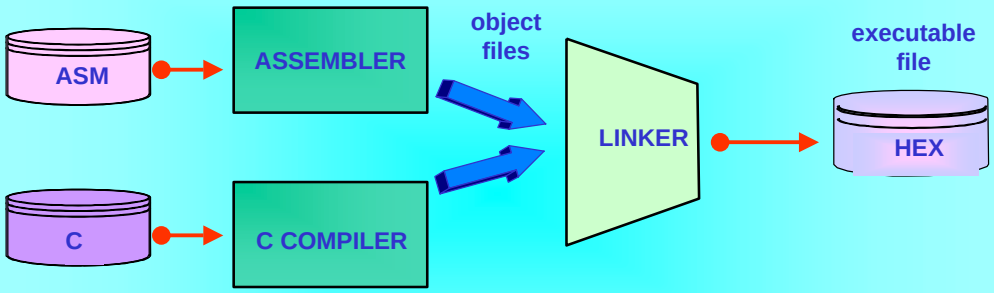
ST62 DEVELOPMENT TOOLS

1. RIDE
2. MAST6 ASSEMBLER
3. From AST6/LST6 to RIDE
4. RCST6 C COMPILER
5. CEIBO and SOFTEC emulators

R*IDE* ST6
RC-ST6 C compiler

- Part of the RKIT-ST6 Package
- Fully integrated into R*IDE*
- Ansi C compiler
- Automatic management of ST6 memory architecture
  - ROM banking
  - RAM banking
  - ROM windowing
- Two memory models: small and large

RIDE ST6
Programming tools



FULLY INTEGRATED INTO R*IDE*

RC-ST6
Restrictions to ANSI-C

- **Small rom size:**
  - Functions length limited to 2KB
- **Small ram size:**
  - Arithmetic types: only 8-bit and 16-bit are implemented, no floating
  - A variable is limited in size to 64 bytes
- **6 levels of stack:**
  - Recursivity is forbidden
  - Reentrance from a higher level of interrupt is allowed

RC-ST6
Extensions to ANSI-C

- **Space qualifiers**
  code, data, scode, sdata, sfr
  - **scode and sdata are related to the non banking area**
  - **sfr is related to the space used to address the microcontroller peripherals**
- **GENERIC keyword**
  - **Applies to pointer**
  - **The object is either in code or data space**

RC-ST6
Extensions to ANSI-C

- **AT keyword**
  Syntax: at *address*
  - **Allows the absolute address of a variable or constant to be specified**
- **EXAMPLE**

```
at 0x02 data char var0, var1;          /* var0 at 0x02, var1 at 0x03 */
```

space qualifier

RC-ST6
Extensions to ANSI-C

- **INTERRUPT keyword**
  Syntax: interrupt *vector_number*
  - **Causes the defined function to be interpreted as an interrupt routine.**
  - ***vector_number* must be specified according to the datasheet**
- **EXAMPLE**

```
void it_timer (void) interrupt   1   /* timer interrupt subroutine is mapped    */
{....}                               /* on interrupt vector 1                   */
```

RC-ST6
Extensions to ANSI-C

- **ASM keyword**
  Syntax: asm *{opcode}*
  - **Allows** hexadecimal code **to be placed at the current address of the executed code.**
  - **Limited in-line assembly**

- **EXAMPLE**

```
void main ()
{ ....
asm{0x6D};                    /* STOP instruction */
...
}
```

RC-ST6
Memory models

- **SMALL model**
  - For devices with up to 4k Rom and 128 bytes Ram
- **LARGE model**
  - For devices with up to 8k Rom and 192 bytes Ram
- **EEPROM**
  - Access to EEPROM will be managed through dedicated subroutines
- **The model is chosen thanks to a menu in R*IDE* (Options/Project/RCST6)**

RC-ST6
Concept of Module

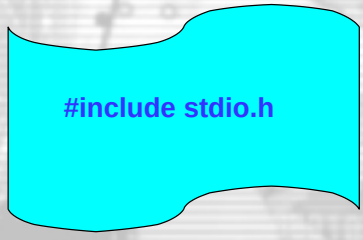
- **APPLIES TO THE LARGE MODEL**
- **WHAT IS A MODULE:**
  - **A module is defined by a couple (DRBR, PRPR)**
  - **Functions and declarations from one source file**
  - **The module identity (DRBR, PRPR) is saved then restores if an interrupt occurs**

RC-ST6
Parameters and local variables

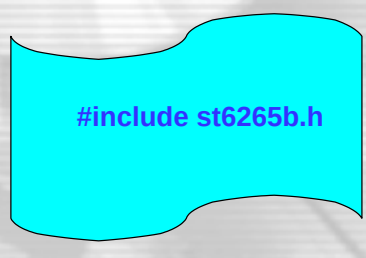
- Stored in a data segment
- Segment is overlayable and relocatable by the linker

RC-ST6
Configuration files / librairies

- RCST6 is delivered with ANSI C Librairies

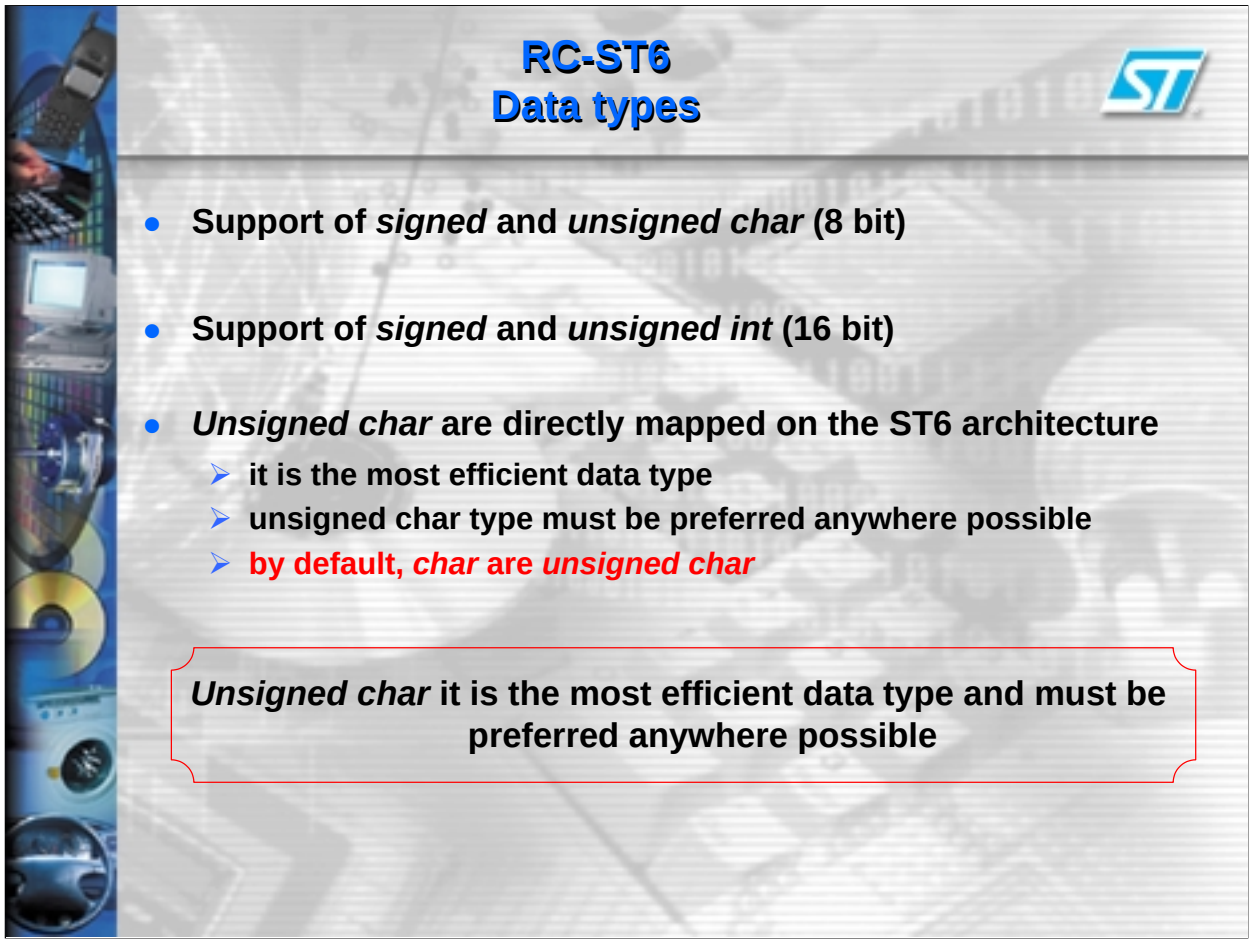


- Specific ST6 configuration files



RC-ST6
Startup file

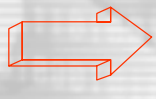
- Initializes DRBR and/or PRPR Registers if any
- Clear ram space
- Initializes global variables
- Reti from nmi mode
- Jump to the main routine
- File can be edited by the user


RC-ST6
Data types
Support of signed and unsigned char (8 bit)
Support of signed and unsigned int (16 bit)
Unsigned char are directly mapped on the ST6 architecture
it is the most efficient data type
unsigned char type must be preferred anywhere possible
by default, char are unsigned char
Unsigned char it is the most efficient data type and must be preferred anywhere possible

RC-ST6
Data types : char versus integer

- Comparison between generated code for unsigned char, and unsigned int

```
char i;
   for ( i = 0; i < 50; i++ ) { ...}

inc    VW11     ; (2)
ld A, VW11      ; (2)
cpi    A, 032h  ; (2)
jrnc   _LC_7    ; (1)
jp main_L1      ; (2)
```

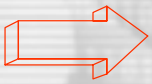


```
int i;
     for ( i = 0; i < 50; i++} { .. };

   ld  A, VW12          ; (2)
   addi   A, 01h        ; (2)
   ld  VW12, A          ; (2)
   jrnc   _LC_155       ; (1)
   inc    VW11          ; (2)
_LC_155:                ;main_LL23 (9)
   ld  A, VW11          ; (2)
   cpi    A, 00h        ; (2)
   jrnz   _LC_156       ; (1)
   ld  A, VW12          ; (2)
   cpi    A, 032h       ; (2)
_LC_156:                ;main_LL24 (3)
   jrnc   _LC_7         ; (1)
   jp  main_L1          ; (2)
```

RC-ST6
Data types : unsigned versus signed

- ***Signed char*** **requires more code than** ***unsigned char***

```
;   if ( j1 < 50 ) f();

LD        A,VW1
CPI       A,0x32
JRNC      main_L1
CALL      _f
main_L1:
```



```
;   if ( SignedChar < 50 ) f();

    LDI       VW0, 0x32
    LD        A, _SignedChar
    CP        A, VW0
    JRR       0x7, VW0,main_LL2
    JRS       0x7, _SignedChar,main_LL3
    JRR       0x7, _SignedChar,main_LL4
main_LL2:
    JRS       0x7, _SignedChar,main_LL4
main_LL3:
    LD      A, _SignedChar
    CP      A, VW0
main_LL4:
```

RC-ST6
Data in ROM

- **Automatic support of data in ROM through ST6 windows mechanism**
- **Automatic control of Data ROM Window Register is generated and optimised by the compiler**
- ***const* qualified global variables are allocated in ROM and accessible through the ST6 ROM window mechanism**
  - **constants limited to the window size of 64 bytes**

example:

```
const char str[ ] = "abcdef";
const char LongTab[100]; /* ERROR */
```

example:

```
const char Const;

char   Char ;
Char = Const;
```



```
ldi  DWR,#window(_Const)
ld   A,#winoffset(_Const)
ld  _Char, A
```

RC-ST6
ROM paging management

- **Optimised automatic mapping of code inside ROM pages**
  - **automatic mapping of function inside ROM pages**
  - **automatic generation of switch code in page 1**
  - **minimise the total number of switch code in page 1**
  - **restriction: functions cannot cross a page boundary**

example:

RAM=0x00-0x3F
page1=0x800-0xF9F

RC-ST6
Interrupt management

- Context is saved and restored
- CONTEXT = A, V, W, X, Y + dedicated RCT6 registers

RC-ST6
Future improvements

- Bit type support
- Specific instructions for bit manipulation
- In-line assembly
- New optimizations


ST62 DEVELOPMENT TOOLS
1. RIDE
2. MAST6 ASSEMBLER
3. From AST6/LST6 to RIDE
4. RCST6 C COMPILER
5. CEIBO and SOFTEC emulators

ST62 EMULATORS
Third parties products

- **SOFTEC**
  - **Two In-circuit real time emulators:**
    - **DS6225A supports ST620x/1x/2x family**
    - **DS6265A supports ST625x/6x family**
  - **Serial port connection to PC**
  - **Ride graphical interface**
  - **Contact:**
    - **Web : http://www.softec.micro.com**
    - **Email : info@softecmicro.com**

ST62 EMULATORS
Third parties products

- **CEIBO**
  - Low cost real time EB-ST62 emulation board
  - Support ST620x/1x/2x/18/28/5x/6x thanks to a base board and 3 Peripheral Emulation Boards
  - Serial port connection to PC
  - RIDE graphical interface
  - Contact
    - Web : http://www.ceibo.com