APPLICATION NOTE

USING THE INDIRECT ADDRESSING MODE WITH ST7

by 8-Bit Micro Application Team

INTRODUCTION

The ST7 assembly language instruction set includes the indirect addressing mode (indexed or not) for short and long variables. The purpose of this document is to show how using the indirect addressing mode is useful.

1 DESCRIPTION OF THE INDIRECT ADDRESSING MODES

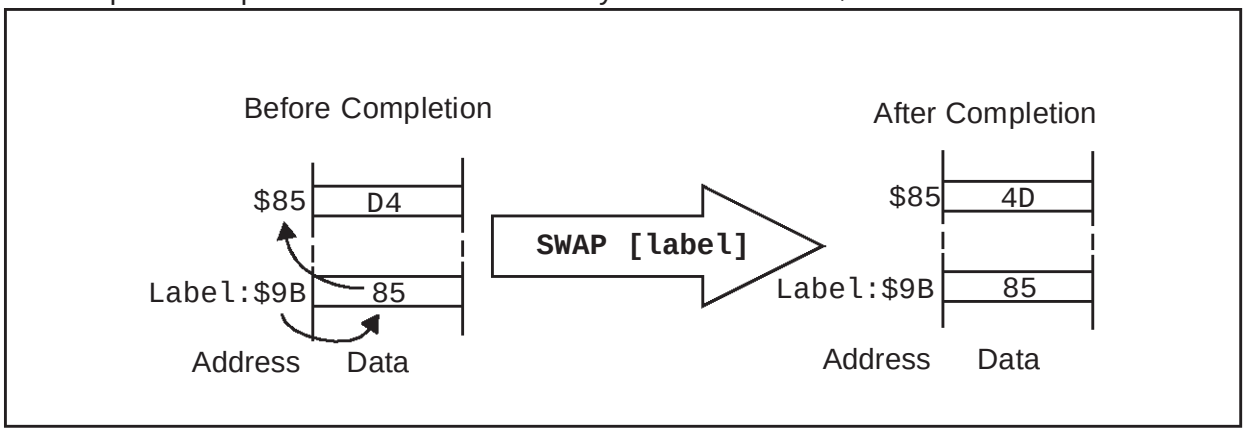
As it is shown in the following examples, the indirect addressing mode is used by the assembly language when the destination byte is inside brackets [ ].

In this mode the label located inside the brackets is called a pointer. In the ST7 instructions set the pointer must be located in the short addressing RAM: its address must be a byte.

1.1 SHORT INDIRECT ADDRESSING MODE

The pointer contains the address of the data to process. This mode allows the result of a previous calculation to be the address of the data to process.

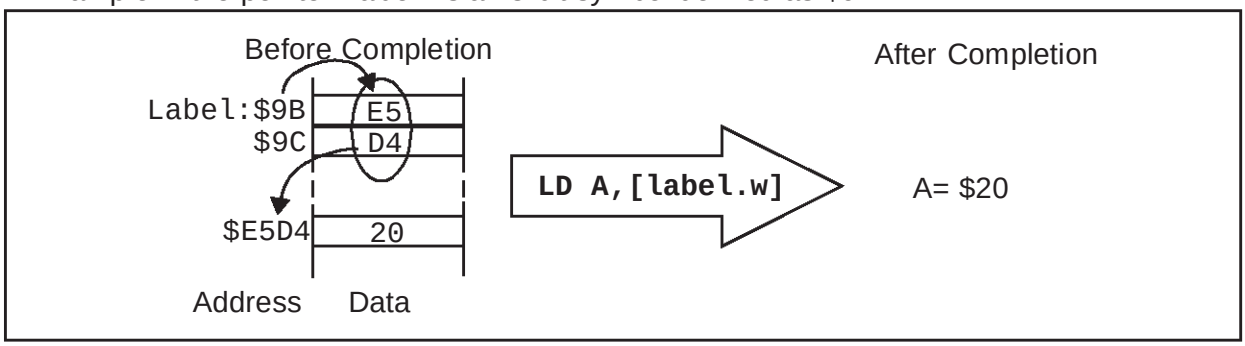
- Addressable space: $00 to $FF.
- Example: if the pointer `label` is an 8 bit symbol defined as $9B.



1.2 LONG INDIRECT ADDRESSING MODE

This mode is similar to the short indirect mode, but the effective address is a 16-bit word. This allows to access the whole address range.

- Addressable space: $0000 to $FFFF.
- Example: if the pointer `label` is an 8 bit symbol defined as $9B.

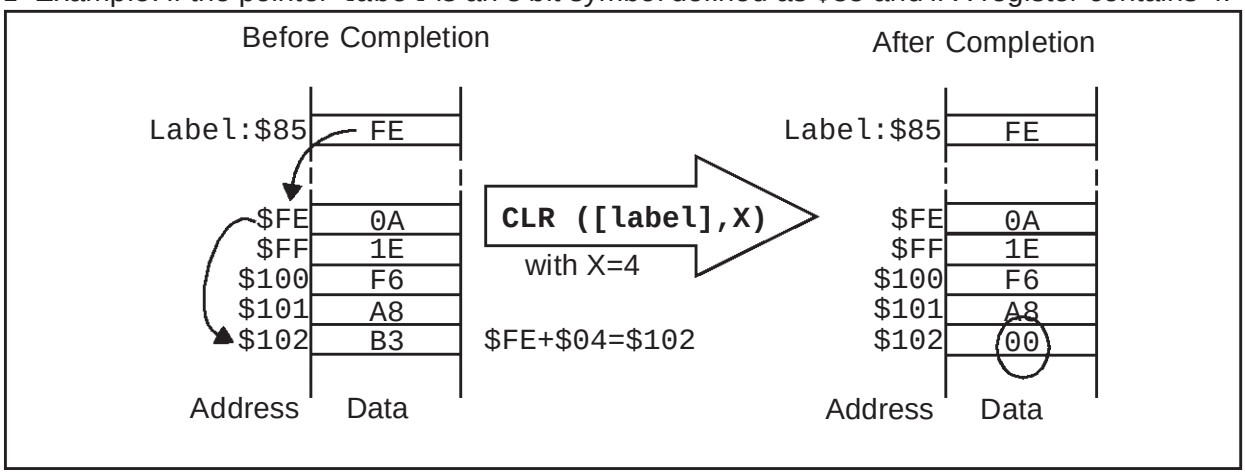


Note the extension ".w" indicates the long indirect addressing mode is used.

1.3 SHORT INDIRECT INDEXED ADDRESSING MODE

In this mode, as the short indirect mode, the byte address following the opcode contains an 8-bit pointer. The pointer content is added with the content of an index register (X or Y). The sum is the effective address of the data to process.

- Addressable space: $000 to $1FE.
- Example: if the pointer `label` is an 8 bit symbol defined as $85 and if X register contains 4.

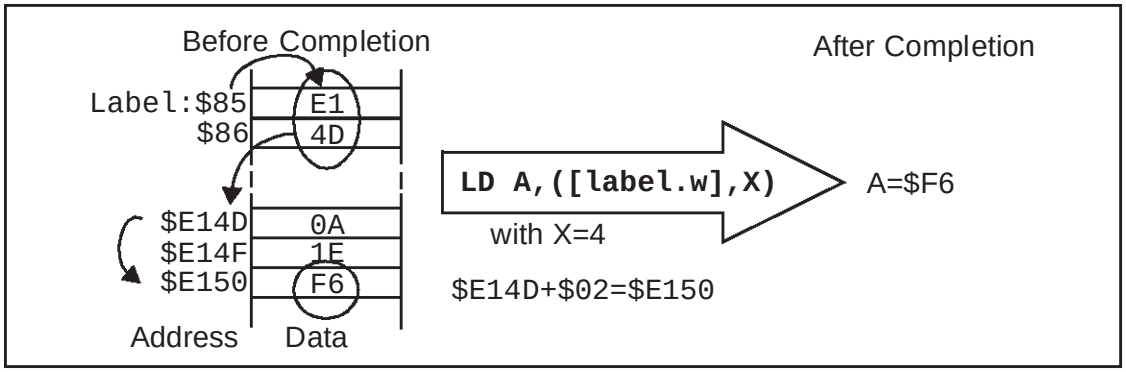


1.4 LONG INDIRECT INDEXED ADDRESSING MODE

This mode uses a 16-bit index in the memory and adds its contents with the 8-bit content of the index register (X or Y), the sum is the effective 16-bit address of the data to process.

Addressable space: $0000 to $FFFF.

Example: if the pointer `label` is an 8 bit symbol defined as $85 and if X register contains 2.



Note the ".W" indicates the long indirect addressing mode is used.

2 ADVANTAGES IN ASSEMBLY LANGUAGE PROGRAMMATION

This addressing mode allows to minimize an application size code. Instead of using a macro to define a repetitive operation with different parameters, the indirect addressing mode permits to use a function.

2.1 EXAMPLE

The purpose of the following example is to write a part of code which increments all the values contained in an array. This code must be reusable in the same program to process different arrays. The first address and the array length are given.

2.1.1 Using a macro and direct addressing mode.

This macro use direct indexed addressing mode.

```
inc_arr MACRO       array, lgth;array is the address of the first byte of
                          ;the array, lght contains the array length
        LOCAL       jump
        LD           X,#{lgth-1}
jump    INC         (array,X),A;direct indexed addressing mode
        LD                DEC   X
        JRPL        jump
        MEND
```

2.1.2 Using a function and indirect instructions

The function uses the indirect indexed mode.

```
.fct_inc LD         ad_array,A     ;ad_array contains the array address,this
                          ;parameter is stored in A, the length in X
.cont   INC         ([ad_array],X));indirect indexed addressing mode
        DEC         X
        JRPL        cont
        RET
```

In this example the incremented array (array1) is defined in the short addressing RAM, `array1` is the starting address and `lght1` the length. The function “fct_inc” is called by the program as it shown in the following lines follow.

```
        LD          A,#array1   ;the array1 address is stored in A
        LD          X,#{lght1-1}
        CALL        fct_inc
```

2.2 CONCLUSION

Every time the macro is used the code is copied in ROM, when using a function the code is written only one time and the data are processed using the indirect addressing mode.

This example shows that only the indirect addressing mode allow to use of a function in this case, thus an optimization in term of application size code using the assembly language.

3 ADVANTAGES IN C LANGUAGE PROGRAMMATION

In many programming applications, the data may have complex forms. To ease the handling of these data, C language is used that simplify the writing of the code by allowing expressions like:

```
AC[k] = AB[j] + AA[i];
```

Where AA, AB and AC are arrays of numbers, and I, J and K the indexes to these arrays. The HIWARE C compiler for ST7 translates this in available machine-language instructions included in the ST7 instruction set.

If these are arrays of bytes which base address is RAM but not in page zero, the following instruction sequence would fit:

```
LD          X, J        ; Set Index register to value of index I of
                        ;array AA
LD          A,([AB.W],X); Get value AA[I] / indirect addressing mode
LD          X, I  ; Set Index register to value of index J of
                        ;array AB
ADD         A,([AA.W],X); Add value of AB[J]/ indirect addressing mode
LD          X, K  ; Set Index register to value of index K of
                        ;array AC
LD          ([AC.W],X),A; Put result into AC[K]
```

In this case, the whole addition is performed in:

- 31 cycles
- 3.875µs at f_{CPU} = 8 MHz
- 15 bytes of code.

If the 3 arrays are located in the page zero (address on 8 bits), the whole addition is performed in:

- 28 cycles
- 3.5µs at f_{CPU} = 8 MHz
- 15 bytes of code.

The obvious compacted size of this generated code is due to the indirect addressing mode in the ST7 instruction set.

This is only one of the many examples where powerful addressing modes help translate high-level language with a good efficiency.

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.