APPLICATION NOTE

ST7 Benefits versus Industry Standard

by 8-Bit Micro Application

INTRODUCTION

The purpose of this note is to present the main advantages of the ST7 core faced with the corresponding industry standard core in term of:

- price (ROM / RAM code size reduction)
- speed (execution time reduction)
- flexibility (more interrupt vectors)

1 GENERAL APPROACH

The following listed ST7 core features allow a reduction of the memory size (ROM and RAM) and of the execution time compared to the industry standard where they are not implemented.

- Y Index register
- Indirect memory access mode
- Stack pointer access
- PUSH / POP instructions
- SWAP instruction
- Up to 16 interrupt vectors

Concerning some features such as the indirect memory access the industry standard do not give alternative to realise such functionality.

The next paragraphs describe one by one the previous pointed out features. When it is possible, a comparison with the industry standard code is done to illustrate the ST7 advantages.

2 Y INDEX REGISTER

The Y register of the ST7 is a duplication of the X index register. All instructions available in the industry standard which use the X register are also available for the Y register in the ST7 instruction set. Two additional ST7 instructions consist of X,Y transfers (LD X,Y and LD Y,X).

Generally, the Y register instructions opcodes are built with one more byte and the execution time with one more cycle than for the X register.

The main application advantages of the Y index register is based on double buffer management and temporary variable storage. The following examples illustrate these advantages.

DOUBLE BUFFER COPY

This small application consists in copying 20 bytes from a `tabx` buffer to a `taby` buffer with a data order inversion: `taby[19-i]=tabx[i]`. The software application details are given in Figure 1. and shows how to use the Y register as a table index like the X register.

Figure 1. Double Buffer Copy

```
.TabCopyCLRX
      LDY,#19
.Loop LDA,(tabx,X)
      LD(taby,Y),A
      INCX
      DECY
```

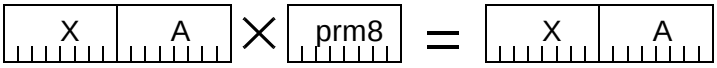
```
.TabCopyLDX,#19
      LDtmp1,X
      CLRX
.Loop LDA,(tabx,X)
      INCX
      LDtmp2,X
      LDX,tmp1
      LD(taby,X),A
      DECX
      LDtmp1,X
```

ST7		Industry Standard Like
14	**ROM size [number of bytes]**	19 (~+36%)
0	**RAM size [number of bytes]**	2
406	**Execution time [number of cycles]**	589 (~+45%)

The analysis of a comparison result highlights the optimization gains against the industry standard code in term of memory size (ROM/RAM) and execution time.

UNSIGNED 8X16-BITS MULTIPLICATION

This small application consists in the unsigned multiplication of a 16-bit value by a 8-bit variable. The 16-bit operand is given by the X (high byte) and A (low byte) registers while the 8-bit one is given by a short memory variable (`prm8`). The calculated result is truncated to 16 bits and returns in X and A registers as described before.



To optimize the ST7 code, the Y register is used as a temporary variable, operand and stack handling interface (see Figure 2.).

Concerning the stack handling management more details are given in Section 4.

Figure 2. Unsigned 8x16-bits Multiplication

```
.Mul16x8LDY,prm8
      MULY,A
      PUSHA
      LD tmp1,Y
      LD A,prm8
      MULX,A
      ADDA,tmp1
      LD X,A
```

```
.Mul16x8LDtmp1,X
      LD X,prm8
      MULX,A
      LD tmp2,A
      LD tmp3,X
      LD X,prm8
      LD A,tmp1
      MULX,A
      ADDA,tmp3
      LD X,A
```

ST7		Industry Standard Like
16	**ROM size [number of bytes]**	19 (~+19%)
2 (+1 in stack)	**RAM size [number of bytes]**	4
48	**Execution time [number of cycles]**	51 (~+6%)

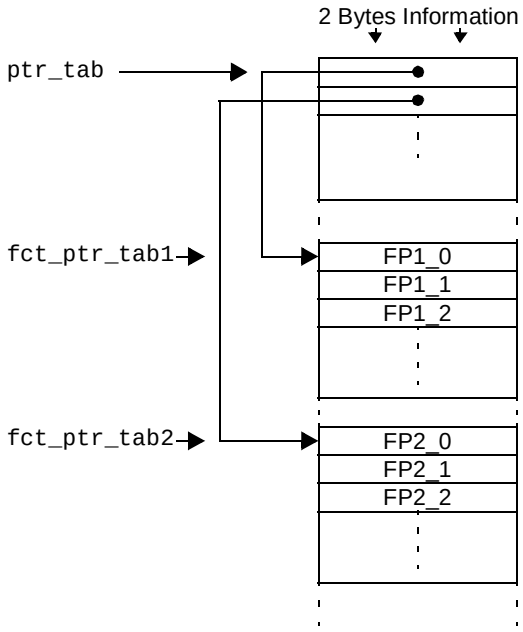
The analysis of a comparison result highlights the optimization gains against the industry standard code in term of memory size (ROM/RAM) and execution time.

3 INDIRECT MEMORY ACCESS MODE

This mode allows to address an area greater than 256 bytes as it is not possible with the indexed mode. It is usually used in table and pointer chain management.

A two dimension function pointer access application example is described in Figure 3. This application points out the advantages of the indirect addressing mode combined with the indexed mode. With only the X register information content and two 16-bit temporary variables (`fct_ptr_tab` and `fct_ptr`), the application jumps to a selected function address.

Figure 3. Two Dimension Function Table Access



X register contains the index of the selected function pointer table in the pointer table.

```
SLLX
LDA,(ptr_tab,X)
LDfct_ptr_tab,A
INCX
LDA,(ptr_tab,X)
LDfct_ptr_tab:1,A
LDX,fct_idx
SLLX
LDA,([fct_ptr_tab.w],X)
LDfct_ptr,A
INCX
LDA,([fct_ptr_tab.w],X)
LDfct_ptr:1,A
```

A jump to the selected function is computed.

This mode is not available in the industry standard instruction set and it is impossible to emulate this mode through standard code.

Please refer to the “AN986 Application Note” for more details about the indirect addressing mode.

4 STACK HANDLING

The ST7 instruction set includes the stack management while the industry standard do not. This part of the instruction set allows code size reduction and gives the possibility to modify the stack for specific applications (such as multi-task kernel...).

The stack management is based on:

- PUSH/POP instructions: push or pop a selected CPU register in the stack.
- LD register transfer instructions: load/store value from/to stack pointer register.

The following examples illustrate the stack management advantages.

Note: When the Y register is used in the main execution tree program, each interrupt routine which uses this Y register has to contains at the first beginning the storage of the register and a restore at the end. These actions can be done through the PUSH Y and POP Y instructions.

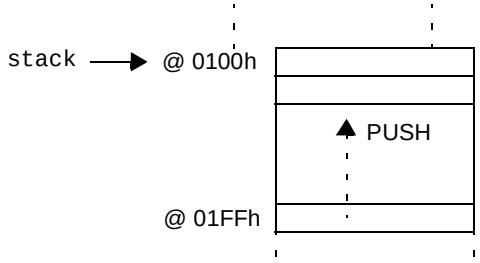
STACK CONTENT DUMP

The application example given by the Figure 4. describes a simple software which read the used stack content and copy it in a standard RAM table (`tab`). This information can also give dynamically the stack depth.

Figure 4. Stack Content Dump

The hardware lowest stack address is given by “stack” variable and the stack depth is 256 bytes.

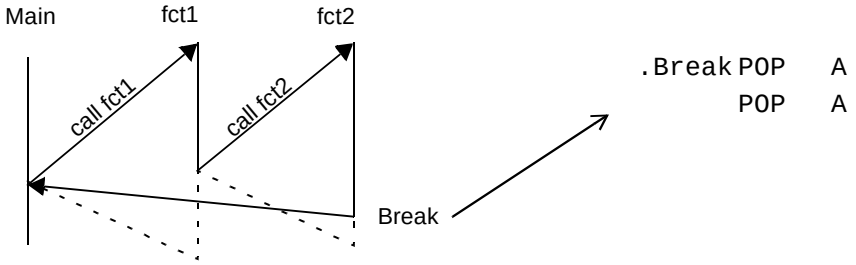
```
.DumpStackLDX,S
.loop  INC   X
       JREQ  End
       LD    A,(stack,X)
       LD    (tab,X),A
       JRT   loop
```



MULTILEVEL FUNCTION BREAK

The application example given by the Figure 5. describes a possibility to generate a multilevel function break (or return). This piece of software can be classed as the C language `break` instruction in a `switch/case` structure.

Figure 5. Multilevel Function Break Management

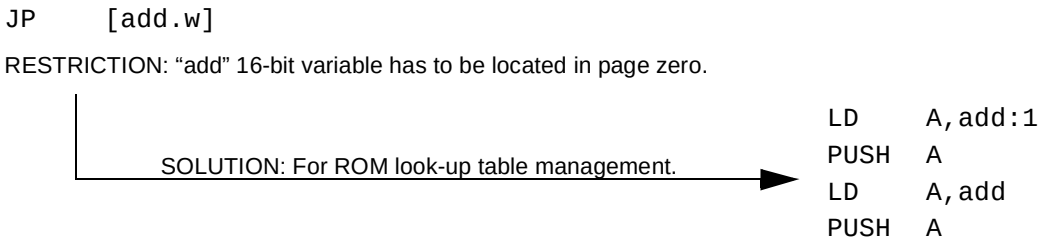



```
.Break POP   A
       POP   A
```

INDIRECT BRANCH THROUGH STACK

As the ST7 indirect addressing mode is always based on variable located in the page zero, a preliminary copy in a temporary page zero variable is often needed. The application example shown in Figure 6. gives an optimum solution for an indirect jump with the indirect variable outside page zero using the stack capability.

Figure 6. Indirect Branch Through Stack



5 SWAP INSTRUCTION

The ST7 instruction set includes a SWAP nibbles instruction while the industry standard do not. This instruction allows code size reduction in bit shift operation and 4-bit field handling in a byte variable.

MORE THAN 3 BIT RIGHT SHIFT

When more than 3 bit right shift of a data byte has to be computed, the optimum code is based on a SWAP instruction instead of a SRL instruction sequence.

The application example given by the Figure 7. describes a 4 bit right shift of the A register.

Figure 7. 4 Bit Right Shift of the A Register

Table 1.

ST7		Industry Standard Like
3	**ROM size [number of bytes]**	4 (~+33%)
5	**Execution time [number of cycles]**	12 (+140%)

The analysis of a comparison result highlights a major optimization gain against the industry standard in term of execution time. The memory size (ROM/RAM) is also significantly reduced.

6 INTERRUPT VECTORS

The ST7 core is able to manage up to 16 interrupt vectors including the RESET and the TRAP when the industry standard is limited to 8 vectors.

This upgrade allows:

- more than 6 peripheral interrupt sources (n x external, n x timer, spi, sci, i2c...)
- more flexible interrupt handling (a minimum interrupt flag source number by vector allows a software test reduction in the interrupt routines)

"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.