Center alignment of a sentence (ASCII string) in a memory field of given character length

Using 8051 micro-controller

Task Description:

Center alignment of a sentence (ASCII string) in a memory field of given character length. The size of the field is an input parameter. Use space characters (ASCII code: 0x20) to fill the empty positions. The field which contains the aligned text does not need to be null terminated.

Inputs:

- Start address of the (null-terminated) string (pointer)
- start address of the field (pointer)
- length of the field (value)

Output: The correctly filled field containing the aligned text

Challenges: Such a code should be written that can handle strings having no character, full string, when a string has only one character less than the memory field or more characters in the string than memory field size. Any of these situations can appear.

Assembly Code Description:

At the beginning of the code symbols FIELD_ADDR_IRAM , FIELD_LEN , CHARACTER for starting are created for starting address of field in RAM, length of the field, and character for alignment respectively. After that, the string is defined at a specific address and terminated with null '0' character, followed by interrupt jump table. In the MAIN function after moving the field length , field starting address and string starting address(string is in code memory), the function STR_ALIGN_CENTER is called inside an infinite loop.

STR_ALIGN_CENTER:

This subroutine contains further 6 subroutines:

- **STR_LENGTH** : Finds number of characters in the string

```
STR_LENGTH:
inc R4                  ;Used to move data in Accumulator for index addrressing to extract characers from DPTR
MOV A,R4                ;Moving value into Accumulator to extract characters via indexed addressing
movc A,@A+DPTR          ;Indexed addressing using Accumulator register
inc R1                  ;counting the string length character by character
jnz STR_LENGTH          ;the loop stops when Ø(null character) is detected

MOV A, R6               ;To move starting address of string in code memory to RØ through A
MOV RØ,A                ;To move starting address of string in code memory to RØ through A
```

- **NORMAL_STR**: Checks if a string is null string or full string or the number of characters in string is one less than the field length, and jumps to their respective subroutines. If not any of the above, then it is a normal string, then it finds the number of empty positions, and moves to the next subroutine in the code to start alignment.
  NORMAL_STR has further 3 subroutines PRE_STR_SPACES, STR_CPY, POST_STR_SPACES. These subroutines will be described later.

- **FULL_STR**: Copies string characters from code memory to data memory using index addressing for a string having same length as memory field, and jumps to LAST subroutine.

```
FULL_STR:
MOV R4,#ØxFF
LOOP4:              ;Subroutine to copy string characters into the memory field excluding null termination character
inc R4              ;Used to move data in Accumulator for index addressing to extract characers from DPTR(string)
MOV A,R4            ;Moving value into Accumulator to extract characters via indexed addressing
movc A,@A+DPTR      ;Moving data at string memory addresses in code memory to accumulator using index addressing
MOV @RØ,A           ;Moving string characters excluding null termination character to given memory field adresses in the internal memory
inc RØ              ;Inceremenring RØ to point at next empty position address in the field memory
djnz R5,LOOP4       ;Number of loop iterations equals number of characters in the string or string length excluding null termination character

JMP LAST            ;jumps to RET in subroutine LAST

; -------------------------------------------------------------------
```

- **ONE_EMPTY_POS**: Aligns string characters for a string having no. of characters one less than the memory field length and jumps to LAST subroutine.

```
ONE_EMPTY_POS:
MOV R4,#ØxFF        ;R4 Will be used later in STR_COPY for assisting A in indexed addressing

STR_COPY2:          ;Subroutine to copy string characters into the memory field excluding null termination character
inc R4              ;Used to move data in Accumulator for index addrressing to extract characers from DPTR(string)
MOV A,R4            ;Moving value into Accumulator to extract characters via indexed addressing
movc A,@A+DPTR      ;Moving data at string memory addresses in code memory to accumulator using index addressing
MOV @RØ,A           ;Moving string characters excluding null termination character to given memory field adresses in the internal memory
inc RØ              ;Inceremenring RØ to point at next empty position address in the field memory
djnz R5,STR_COPY2   ;Number of loop iterations equals number of characters in the string or string length excluding null termination character
```

- **DEFAULT**: Fills the whole memory field with space characters for a null string(having no characters) or string having more characters than memory field

```
DEFAULT:
MOV A,#CHARACTER        ;The alignment character is moved to accumulator
LOOP1:
      MOV @RØ,A         ;The alignment character is moved to the address pointed by RØ
      inc RØ            ;incrementing string address
      DJNZ R7,LOOP1     ;Loop iterations are equal to field length
MOV A,R2                ;Field length goes to R7 register
MOV R7,A                ;Field Length stored in a safety register to  avoid infinte iterations of LOOP1
JMP LAST                ;jumps to RET in subroutine LAST
```

- **LAST**: It is used to Return the main subroutine STR_ALIGN_CENTER and infinite LOOP starts

```
LAST:               ;It is used to Return the main subroutine STR_ALIGN_CENTER and infinite LOOP starts
RET                 ;Return
END                 ; End of the source file
```

Subroutines in NORMAL_STR:

- **PRE_STR_SPACES**: Subroutine for putting space characters behind the string in data memory field.

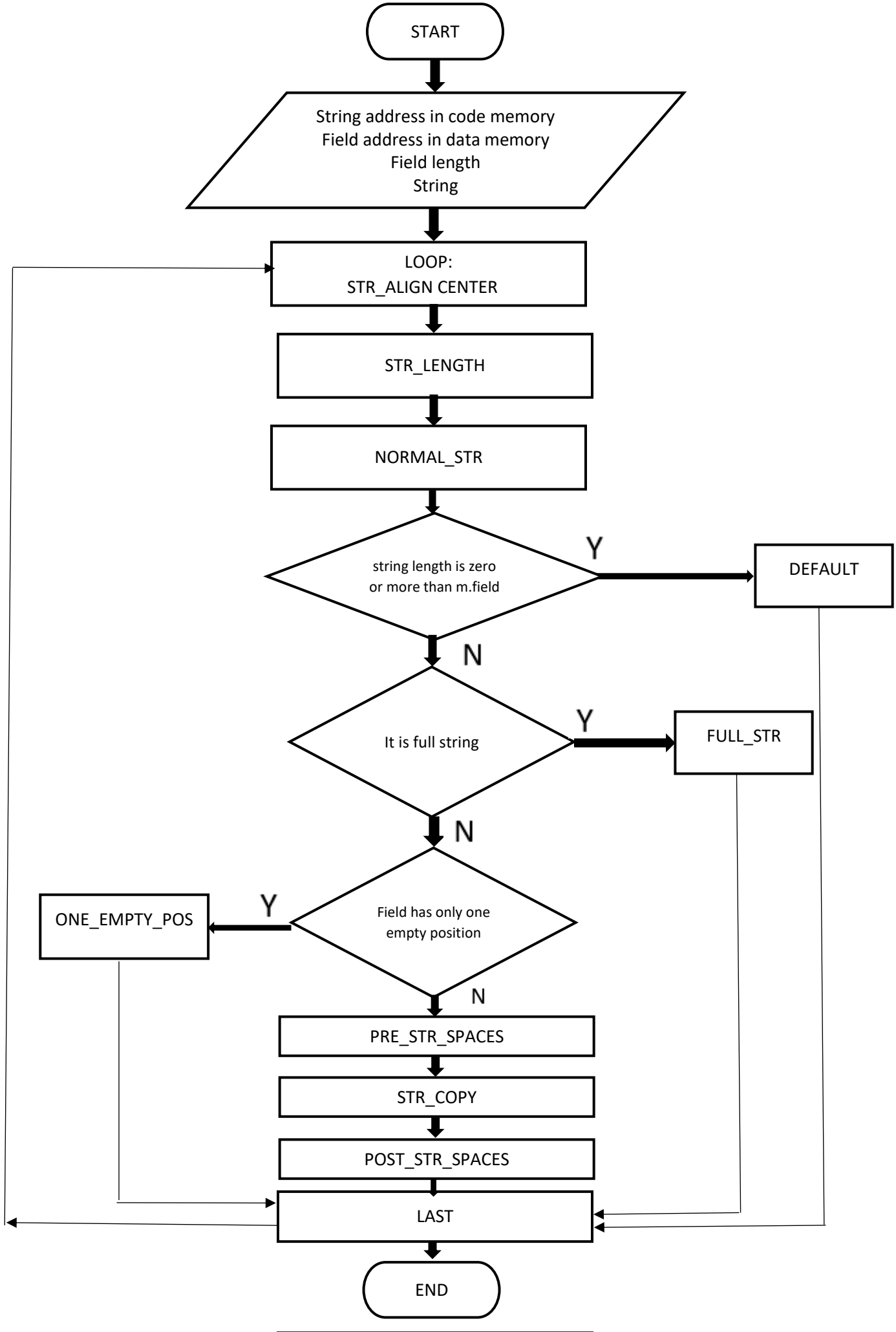
```
PRE_STR_SPACES:          ;Subroutine for putting space characters behing the string in data memory field
MOV A,#CHARACTER         ;Moving ASCII code of space character into accumulator
MOV @RØ,A                ;Placing space characters at the addresses of memory field pointed by RØ
inc RØ                   ;Moving to the address of next position in memory field
djnz R1,PRE_STR_SPACES   ;Number of iterations of loop is equal to number of space characters to be placed

MOV R4,#ØxFF             ;R4 Will be used later in STR_COPY for assisting A in indexed addressing
```

- **STR_COPY:** Subroutine to copy string characters into the memory field excluding null termination character.

```
STR_COPY:          ;Subroutine to copy string characters into the memory field excluding null termination character
inc R4             ;Used to move data in Accumulator for index addressing to extract characers from DPTR(string)
MOV A,R4           ;Moving value into Accumulator to extract characters via indexed addressing
movc A,@A+DPTR     ;Moving data at string memory addresses in code memory to accumulator using index addressing
MOV @RØ,A          ;Moving string characters excluding null termination character to given memory field adresses in the internal memory
inc RØ             ;Inceremented RØ to point at next empty position address in the field memory
djnz R5,STR_COPY   ;Number of loop iterations equals number of characters in the string or string length excluding null termination character
```

- **POST_STR_SPACES:** Subroutine to Subroutine for putting space characters ahead/after the string.

```
POST_STR_SPACES:          ;Subroutine to Subroutine for putting space characters ahead/after the string
MOV A,#CHARACTER          ;Moving ASCII code of space character into accumulator
MOV @RØ,A                 ;Placing space characters at the addresses of memory field pointed by RØ
inc RØ                    ;Moving to the address of next position in memory field
djnz R2,POST_STR_SPACES   ;Number of iterations of loop is equal to number of space characters to be placed
JMP LAST                  ;jumps to RET in subroutine LAST
```


START
String address in code memory
Field address in data memory
Field length
String
LOOP:
STR_ALIGN CENTER
STR_LENGTH
NORMAL_STR
string length is zero
or more than m.field
Y
DEFAULT
N
It is full string
Y
FULL_STR
N
Field has only one
empty position
Y
ONE_EMPTY_POS
N
PRE_STR_SPACES
STR_COPY
POST_STR_SPACES
LAST
END