GETTING STARTED WITH THE TMS320F24x PROCESSOR

Tutorial 4: Using Direct Addressing

New Instructions Introduced

```
LDP
SACL        SACH
SPLK
```

New Flag Introduced

```
CNF
```

Introduction

Most programs need to store variables. For this purpose, the TMS320F24x provides data memory (RAM) on the chip. This tutorial looks at ways of storing values in data memory and retrieving them.

Data Types

In C, we declare variables as follows:

Example 4-1.

```
unsigned char ch; /* 8-bit variable */
unsigned int i;   /* 16-bit variable */
unsigned long l;  /* 32-bit variable */
```

We can use all these types of variables with the TMS320F24x. However, the architecture of the TMS320F24x uses 16-bit data widths, so for small variables we would tend to work in terms of `int` rather than `char`.

Loading the Accumulator from Data Memory

To copy a 16-bit value from a data memory address to the accumulator we use the instruction `LACC` (load accumulator). This is shown in Example 4-2.

Example 4-2.

```
        LACC 70h    ; Load the accumulator with the 16-bit
                    ; contents of data memory address 70h.
                    ; The contents of data memory address
                    ; 70h remain unchanged.
```

This method of moving data to and from data memory is known as *direct addressing*. The location in data memory is given directly by the operand. In this context, the word *load* can also be thought of as *copy*.

Suppose that data memory address 70h contains the value 3456h. After executing the instruction `LACC 70h`, the accumulator will contain 00003456h. The contents of data memory address 70h will remain unaffected.

The single operand used with the instruction `LACC` gives the address in data memory of the variable to be used. There is the restriction that the operand must lie between 0 and 7Fh (0 and 127 decimal). Note that we do not use the symbol # before the address.

The following are all incorrect attempts to load the accumulator with the contents of a data memory address:

Example 4-3.

```
        LACC 80h    ; Incorrect. Operand out of range.
                    ; Maximum allowed value is 7Fh (127).
        LACC -1h    ; Incorrect. Operand must be positive.
        LACC #40h   ; Incorrect addressing mode. Load the
                    ; accumulator with the number 40h rather
                    ; than the contents of data memory
                    ; address 40h.
```

When loading the accumulator from data memory, we can write the code using two different syntaxes.

Example 4-4.

```
        LACC 21h    ; First syntax. Load the accumulator with
                    ; the contents of data memory address
                    ; 21h.
        LACC @21h   ; Second syntax. Load the accumulator
                    ; with the contents of data memory
                    ; address 21h.
```

We can use either syntax. The first syntax is the one used in the Texas Instruments data books and for consistency, this is the syntax we shall use in these tutorials.

Setting the Data Page

Because the operand used with the instruction `LACC` can only lie in the range 0 to 127, how do we use direct addressing to access the full 544 words of data memory?

The data memory of the TMS320F24x is organized into *pages*. By *page* we mean that the data memory is subdivided into sections of fixed size, rather like a book. This means that to refer to a particular data memory address, we must specify the page number.

The structure of the data memory of the TMS320F243 is shown in Table 4-1.

Table 4-1. TMS320F243 Memory Blocks

Block Name	Data Page	Addresses	Notes
B2	0	60h to 7Fh	Reserved.
	1	80h to FFh	Do not use
	2	100h to 17Fh	Reserved
	3	180h to 1FFh	
B0	4	200h to 27Fh	Available when
	5	280h to 2FFh	CNF = 0
B1	6	300h to 37Fh	Always available
	7	380h to 3FFh	

The operand used with the instruction `LACC` provides only the lowest 7 bits of the data memory address. The other 9 bits come from another source known as the *data page pointer* (`DP`).

In order to work on the contents of a particular data memory address, we must first set the data page number. To do so we use the instruction `LDP` (load data page)

Example 4-5.

```
        LDP #4h     ; Select data page 4. Gain access to
                    ; data memory addresses 200h to 27Fh.
```

The instruction `LDP` takes a single operand, which is the page number. The maximum value allowed by the assembler is 511. For the TMS320F243 we use a value in the range 0 to 7 to address the elements of the data memory.

The following examples are all incorrect attempts to load the data page pointer with a specific value.

Example 4-6.

```
        LDP #-2h    ; Incorrect. Operand must be positive.
        LDP 5h      ; Incorrect. Will load the value from
                    ; data memory address 5h, which is
                    ; probably not 5h.
        LDP #512    ; Incorrect. Operand out of range.
                    ; The maximum allowed value is 511.
```

Once the instruction LDP has been executed, the data page (DP) remains the same until another instruction is executed, which alters the data page

Reserved Data Memory Addresses

In order to address the full 554 words of data memory on the TMS320F243, we use values of DP between 0 and 7. However, on the TMS320F243 DSK, not all the data pages are available for general use. Read and writes outside this range will not be using data memory and will therefore be unreliable.

In the TMS320F243 databook, on Page 0, only addresses 60h to 7Fh are available for general purpose. The data memory addresses 0 to 5Fh are reserved for the memory-mapped registers. However, on the TMS320F243 DSK, Page 0 is not available at all; every location is reserved.

Furthermore, the data memory addresses on Page 1 of the TMS320F243 DSK are always unavailable. It is therefore incorrect to write:

Example 4-7.

```
        LDP #1      ; Incorrect attempt to select Page 1.
                    ; This page is not available on the
                    ; TMS320F243 DSK for general purpose.
```

Configuring Data Memory

The TMS320F243 has a Harvard architecture, which means that program memory (ROM) is kept separate from data memory (RAM). There is, however, one exception to this rule; the data block B0 can be mapped into either program memory or data memory. This is in order to carry out certain multiplication instructions.

The power-up default for the TMS320F243 DSK is to put data pages 4 and 5 into program memory. This makes data block B0 unavailable for use with instructions such as LACC. Any write to pages 4 or 5 will be unsuccessful; a read from one of these pages will be unreliable.

Mapping of these data blocks is under the control of the *configuration bit* (CNF). When the value of CNF is 0, then data block B0 is available for general purpose. On the other hand, when CNF is 1, the data block B0 is mapped into program memory.

To change the value of CNF we use the instructions CLRC (clear control bit) and SETC (set control bit).

Example 4-8.

```
                CLRC CNF    ; CNF = 0. Data block B0 is
                            ; mapped into data memory for general
                            ; purpose use.
                SETC CNF    ; CNF = 1. Data block B0 is
                            ; mapped into program memory. It is
                            ; not available for use as general
                            ; purpose memory.
```

In Example 4-8, the operand `CNF` is used.

Copying From Data Memory to the Accumulator

We have already seen how to load the accumulator from data memory. However, we have conveniently ignored the *data page pointer* (`DP`)!

In order to copy a value from a data memory address to the accumulator, we need two instructions. The first instruction sets the data page and the second instruction provides the remaining part of the address.

To load the accumulator with the contents of data memory address 320h, from Table 4-1 it can be seen that we need to use Page 6. This gives us access to data memory addresses 300h to 37Fh.

Example 4-9.

```
                LDP #6h     ; Data page 6. Gain access to data memory
                            ; addresses 300h to 37Fh.
                LACC 20h    ; Load the accumulator with the contents
                            ; of data memory address 300h + 20h =
                            ; 320h.
```

Copying from the Accumulator to Data Memory

So far we have loaded a value from data memory into the accumulator. We can also move data in the opposite direction.

Suppose the accumulator already contains 12345678h. To copy the low word of the accumulator (i.e. 5678h) to data memory address 211h we can write:

Example 4-10.

```
                CLRC CNF    ; Configure block B0 as data memory.
                LDP #4h     ; Data page 4. Gain access to data memory
                            ; addresses 200h to 27Fh in Block B0.
                SACL 11h    ; Store the low word of the accumulator
                            ; at data memory address 200h + 11h =
                            ; 211h. This data memory address now
                            ; contains 5678h.
```

The instruction SACL (store low accumulator) takes a single 7-bit operand, which is the address in data memory. The *data page pointer* (DP) provides the remaining 9 bits of the address.

In a similar way, we can copy the high 16 bits of the accumulator to data memory address 252h using the instruction SACH (store accumulator high). If the accumulator already contains 12345678h then:

Example 4-11.

```
        CLRC CNF        ; CNF = 0. Block B0 into data
                        ; memory for general purpose.
        LDP #4h         ; Data page 4. Gain access to data memory
                        ; addresses 200h to 27Fh.
        SACH 52h        ; Store high word of accumulator at data
                        ; memory address 200h + 52h = 252h. This
                        ; data memory address now contains 1234h.
```

With the instructions SACL and SACH, the word *low* means right-most 16 bits and the word *high* means left-most 16 bits.

Saving the Accumulator in Data Memory

In order to copy the entire contents of the accumulator to data memory, we must use both the instructions SACL and SACH. Example 4-12 copies the contents of the accumulator to data memory addresses 300h and 301h.

Example 4-12.

```
        .setsect        ".text", 8800h

start:  LDP #6h         ; Data page 6. Gain access to data
                        ; memory addresses 300h to 37Fh.
        SETC SXM        ; Turn on sign-extension mode.
        LACC #0FEDCh    ; Load accumulator with the known
                        ; test value FFFFFEDCh.
        SACH 0h         ; Store the high word of the
                        ; accumulator at data memory address
                        ; 300h + 0h = 300h. This data memory
                        ; address now contains FFFFh.
        SACL 1h         ; Store low word of accumulator at
                        ; data memory address 300h + 1h =
                        ; 301h. This data memory address now
                        ; contains FEDCh.
        B start         ; Go round again.
```

Putting a Value into Data Memory

How would we store the value 4321h at data memory address 2C4h?

One way would be to load the accumulator with 4321h and then store this value at data memory address 280h + 44h = 2C4h.

Example 4-13.

```
        CLRC CNF        ; CNF = 0. Map block B0 into data
                        ; memory for general purpose.
        LACC #4321h     ; Load the accumulator the with the
                        ; constant 4321h. The accumulator now
                        ; contains 00004321h.
        LDP #5          ; Data page 5. Gain access to data memory
                        ; addresses 280h to 2FFh.
        SACL 44h        ; Store the low word of the accumulator at
                        ; data memory address 280h + 44h = 2C4h.
                        ; This data memory address now contains
                        ; 4321h.
```

A neater way is to use the instruction `SPLK` (store long-immediate value to data memory).

Example 4-14.

```
        CLRC CNF        ; CNF = 0. Map block B0 into
                        ; data memory for general purpose.
        LDP #5h         ; Data page 5. Gain access to data
                        ; memory addresses 280h to 2FFh.
        SPLK #4321h, 44h ; Store the constant 4321h at data
                        ; memory address 280h + 44h = 2C4h.
                        ; This data memory address now contains
                        ; 4321h.
```

The Difference Between Load and Store Instructions

So far we have used both *load* and *store* instructions.

A *load* instruction puts a value into the accumulator. The value can be a number or the contents of a data memory address.

Example 4-15.

```
        LDP #7          ; Data page 7. Gain access to data memory
                        ; addresses 380h to 3FFh.
        LACC #0348h     ; Load the accumulator with the constant
                        ; 348h. The accumulator now contains
                        ; 00000348h.
        LACC 70h        ; Load the accumulator the with the
                        ; contents of data memory address 380h +
                        ; 70h = 3F0h.
```

A *store* instruction puts a value into data memory. The value can be a number or the contents of the accumulator.

Example 4-16.

```
        LDP #6              ; Data page 6. Gain access to data
                            ; memory addresses 300h to 37Fh.
        SPLK #1783h, 23h    ; Store the constant 1783h at a data
                            ; memory address 300h + 23h = 323h.
        SACL 52h            ; Store the low (left-most 16 bits)
                            ; of the accumulator at data memory
                            ; address 300h + 52h = 352h.
```

Testing a Data Memory Address

At the start of a program it is good practice to test that the data memory is reliable before using it. So how do we test a particular data memory address to ensure that it is good i.e. it is not reserved or unavailable?

One way is to write a value to a specific data memory address then read back from the same data memory address. The value should not have changed.

Example 4-17.

```
        CLRC CNF            ; CNF = 0. Map data memory block B0
                            ; into data memory for general
                            ; purpose.
        LDP #4h             ; Data page 4. Gain access to data
                            ; memory addresses 200h to 27Fh.
        SPLK #5555h, 20h    ; Store the constant 5555h at data
                            ; memory address 200h + 20h = 220h.
                            ; This data memory address now
                            ; contains 5555h.
        LACC 20h            ; Load value back into the
                            ; accumulator. The accumulator now
                            ; contains 00005555h.
        SPLK #0AAAAh, 20h   ; Store constant AAAAh at data memory
                            ; address 200h + 20h = 220h. This data
                            ; memory address now contains AAAAh.
        LACC 20h            ; The low word of the accumulator
                            ; (right-most part) now contains
                            ; AAAAh.
```

Note that here we have used the numbers 5555h and AAAAh. These are the inverse of each other. To change from 5h (0101 binary) to Ah (1010 binary) requires every bit to be changed to its opposite value.

Testing Multiple Data Memory Addresses

We shall now write values to three different data memory addresses and read them back. To do this we need to change the *data page pointer* for each read / write.

Example 4-18.

```
        .setsect ".text", 8800h

start:  SETC CNF            ; Map RAM block B0
                            ; into program memory.
        LDP #0              ; Data page 0. Gain access to
                            ; data memory addresses 60h to
                            ; 7Fh.
        SPLK #5555h, 65h    ; Attempt to write 5555h to
                            ; data memory address 65h (a
                            ; reserved data memory address).
        LACC 65h            ; Fail. Different value is read
                            ; back.

        LDP #4              ; Data page 4. Gain access to
                            ; data memory addresses 200h to
                            ; 27Fh.
        SPLK #0AAAAh, 22h   ; Attempt to write AAAAh to data
                            ; memory address 200h + 22h =
                            ; 222h.
        LACC 22h            ; Fail. Different value is read
                            ; back.

        LDP #5              ; Data page 5. Gain access to
                            ; data memory addresses 280h to
                            ; 2FFh.
        SPLK #0AA55h, 43h   ; Write value AA55h to data
                            ; memory address 280h + 43h =
                            ; 2C3h.
        LACC 43h            ; Fail. Different value is read
                            ; back.

        LDP #7              ; Data page 7. Gain access to
                            ; data memory addresses 380h to
                            ; 3FFh.
        SPLK #55AAh, 12h    ; Write value 55AAh to data
                            ; memory address 380h + 12h =
                            ; 392h.
        LACC 12h            ; Same value is read back.

        B start             ; Go round again from the
                            ; beginning.
```

TMS320F243 DSK EXPERIMENTS

The experiments shown here have been designed for the TMS320F243. However, they can also be applied to other devices in the TMS320F243 family, although the size of data memory and the available addresses do vary.

Equipment Required

TMS320F243 DSK

Experiment 4-1.

Objective: To copy from the accumulator to a data memory address.

Enter the code from Example 4-12 into a text editor, save it, assemble it and load into the TMS320F243 DSK. Set a data memory window starting at 300h.

Run the program. The accumulator should contain FFFFFEDCh, data memory address 300h should contain FFFFh and data memory address 301h should contain FEDCh.

Experiment 4-2.

Objective: To show the effect of writing to an unavailable data memory address.

Enter the code from Example 4-18 into a text editor, save it, assemble it and load into the TMS320F243 DSK. Setup a memory window to start at address 0, another to start at address 100h and a third to start at address 200h.

Single step through the code and note the effects upon the data memory addresses being worked upon and the accumulator. The read from data Page 0, Page 4 and Page 5 should not be successful.

Experiment 4-3.

Objective: To show the effect of CNF on data memory.

Enter the code from Example 4-18 into a text editor. Change the instruction `SETC CNF` to `CLRC CNF`. Save the program, assemble it and load it into the TMS320F243 DSK. Setup a memory window to start at address 0, another to start at address 270h and a third to start at address 280h.

Single step through the code and note the effect upon the data memory locations and the accumulator.

The read / writes on data Pages 4 and 5 should now be successful.

Experiment 4-4.

Objective: To Determine the value of the data page pointer (DP) at power up.

Load Example 4-12 into a text editor, save, assemble and load into the TMS320F243 DSK, but do not run the program.

Open the CPU Window. Make a note of the value of the data page pointer (DP).

Design Problem 4-1.

Write a program to write one value to a different data memory address on each of data Pages 4, 5, 6 and 7. This will require `CNF` to be correctly configured.

Self-Test Questions..... *Click Here To View* Answers!

1.	The instruction `LACC` means: a) Load accumulator b) Load all components c) Logical AND with carry d) Logical AND and clear carry
2.	What is meant by the term *direct addressing*?
3.	If data memory address 70h contains 3333h, after the instruction `LACC 70h`, what are: a) the contents of data memory address 70h? b) the contents of the accumulator?
4.	Which one of the following instructions is correct? a) `LAC 17` b) `LACC 0h` c) `LACC 128` d) `LACC -1`
5.	The instruction `LDP` means: a) Load Decimal Point b) Load Data Page c) Left Decimal Point d) Long Data Page
6.	What happens if the value of `DP` is out of range?
7.	What effect does the `CNF` flag have upon the data memory of the TMS320F243?
8.	What two instructions are used to change the value of `CNF`?
9.	The instruction `SACL` means: a) Save And Close b) Shift Accumulator Left c) Store Accumulator Low d) Store And Clear
10.	The instruction `SACH` means: a) Save Channel b) Shift Accumulator High c) Store And Change d) Store Accumulator High
11.	Write code to store the full the full 32 bits of the accumulator at data memory addresses 240h and 241h.
12.	The instruction `SPLK` means which of the following? a) Save Pointer to Long Constant b) Shift P register and Link c) Step and Link d) Store Long-immediate value to data memory
13.	What are the differences between *load* and *store* instructions?
14.	Write code to store the value 1234h at data memory address 342h and to store the value 5678h at data memory address 343h.

References

TMS320F/C24x DSP Controllers. CPU and Instruction Set. Reference Number: SPRU160.

TMS320F/C240 DSP Controllers. Peripheral Library and Specific Devices. Reference Number: SPRU161

TMS320F243, TMS320F241 DSP Controllers. Reference Number SPRS064.

CLICK TO VIEW
Tutorials
1 2 3 4 5 6 7 8 9 10

Click here to view..... Route Map