Table Look-up and Interpolation on the TMS320C2xx

Application Report
Literature Number: BPRA046

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Table of Contents

1. Overview

In digital motor control applications, table interpolation is an operation which is always performed, wheel round after round. The rapidity of the table interpolation conditions a correct working of the system, so the DSP should be able to do it as quickly as possible. Saving time for table interpolation will allow more possibilities for other software.

Different kinds of tables may be used: constant or non-constant steps, 2D or 3D, group of abscissa followed by the group of corresponding ordinates, and so on.

In order to help the customer to understand table interpolation, we have in this document presented different solutions for table look-up and interpolation. These solutions should be table size optimized, speed optimized, and precision optimized.

2. Interpolation principle

The general formula for calculating the table interpolation value Y of a number X is:

$$Y = y_i + \underbrace{\frac{X - x_i}{x_{i+1} - x_i}}_{r\ =\ \text{ratio}} (y_{i+1} - y_i)$$

Figure 1: formula

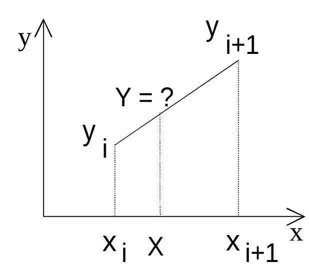


Figure 2: interpolation

where: $\{x_i\}$ = {first coordinates of the table},
$\{y_i\}$ = {second coordinates of the table},
i chosen so that $x_i < X < x_{i+1}$.

Table interpolation can be divided into two steps:

- table look-up: it consists of looking through the whole table in order to find in which interval $[x_i, x_{i+1}[$ the considered point X is located, with $x_i < X < x_{i+1}$.
- interpolation: it consists of realizing the above calculation (see Figure 1: formula) in order to obtain Y.

3. Fixed step table

With a fixed step table, it is possible to have a correspondence between the address of a point in the table and its abscissa. In this way the table look up is instantaneous and constant in execution and the table size is reduced, only ordinates are stored, and abscissa are memory addressed .

Values are uniformly spaced, in this way a simple linear interpolation can be used to compute the value between table entries. The simple linear interpolation uses the values of two consecutive table entries as the end point of a line segment. Sample points for parameters values falling between table entries assume values on the line segment between the points.

Constraint :

- Table must have constant steps.
- Integer power of 2 step abscissa (2^p with p equal from 1 to 16).

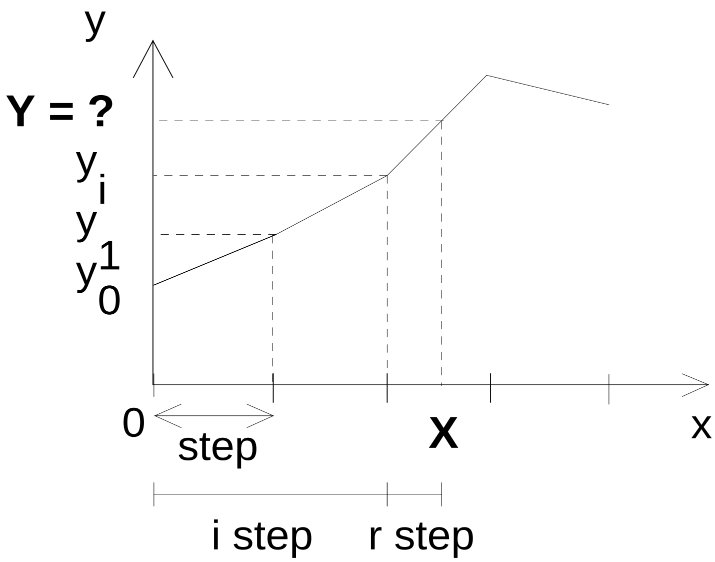
Advantage :

- Easy table Look-up.
- Short table.

Example of a fixed table step table:

0h	y(0)	The ordinate y(0) corresponds to x(0) = 0.
1h	$y(1*2^p)$	The ordinate y(1) corresponds to $x(1) = 1*2^p$.
2h	$y(2*2^p)$	The ordinate y(2) corresponds to $x(2) = 2*2^p$.
3h	$y(3*2^p)$	The ordinate y(3) corresponds to $x(3) = 3*2^p$.
4h	$y(4*2^p)$	The ordinate y(4) corresponds to $x(4) = 4*2^p$.

With this table organization, it is easy to point to the good address. Division from the 2^p value gives the position of the ordinate in the table and the remainder is used for interpolation.



Example:

Suppose p=4,
Each step is 2^4 =16 The table looks like this:

1010h	7
1011h	20
1012h	30
1013h	50

This can be translated into: p=4h, y(0)=7h, y(16)=20h, y(32)=30h, y(48)=50h.

In this way, to find Y(16), we have to divide 16 by 2^4, result is 1, this value is added to beginning of the table 1010h, and the address pointed contains the result.

To find Y(18), we have to divide 16 by 2^4 , the integer result is 1, the remainder, not equal zero is used for interpolation.

The sample TMS320C2xx implementation of this linear interpolation scheme given in Annexe, is an enhancement of the table look-up table. Each time this subroutine is called, the next sample point is calculated.

Here is the main part of this function:

The Y data table organization is :

Begin of the Y table : $Y(0)$
$Y(1*2^p)$
$Y(2*2^p)$
..
..
$Y(n*2^p)$

2^p is the constant abscissa step.

The value to interpolate is Y(Xdata), with $Xdata<n*2^p$

Program :

```
LAC     Xdata,16-d    ;isolate the indice by a 2^p division
SACH    indice        ;fixed position in the table
SACL    reste         ;remainder
LALK    begin_Y_table ;address of beginning of the table
ADD     indice        ;address of the nearest first indice
TBLR    Y1            ;Load Y1 with the ordonate of the
                      nearest 1st indice
ADD     #1
TBLR    Y2            ;Load Y2 with the ordonate of the
                      nearest 2nd indice
LAC     Y2
SUB     Y1            ;difference between the two Y value
SACL    tmp
LT      reste
MPY     tmp
SPH     tmp           ;interpolation between Y1 and Y2
LAC     Y1
ADD     tmp           ;Y(Xdata) in ACCU
```

Processor utilization

Function	Cycles	Execution Time
Interpolation	21	1.05µs

Memory utilization

Function	ROM (words)	Stack levels	Registers used	RAM (words)
Interpolation	17+tables	none	none	5

If this part of s/w is inserted directly in line with the code of a master program, avoiding the overhead of a subroutine, a sample can be computed in only 1.05 microsecond. If the program is used as a subroutine, each sample can be computed in 1.3 microsecond.

An implementation with boundary conditions, abscissa out of range, is given in the Annexe.

4. Generic table Look-up and interpolation in an ordered table

Generic means that there is no particular constraint for the ordered data table. Steps are not constant. Abscissas and corresponding ordinates are present in table. In this case, the look-up is more complex than in the fixed step table.

Example of a generic table:

0h	X(0)	Y(X(0))
1h	X(1)	Y(X(1))
2h	X(2)	Y(X(2))
3h	X(3)	Y(X(3))
4h	X(4)	Y(X(4))
...	....	....

There are many ways to implement the table look-up and interpolation in an ordered table.

- The first method used with small table is to read each abscissa of the table in order to determine in which interval the searched abscissa is located.
- The second method takes advantage of the TMS320C2xx capability of performing bit reversed addressing by proceeding by comparison between the searched abscissa and the middle point of an interval which will be divided by two at each iteration. In this case, we assume that the size of the search

table is some integer power of 2 (2^n). In this case a maximum of n iterations is required to complete the search.

A total solution with the second look-up method and a linear interpolation is presented.

The following function returns the ordinate of the searched abscissa which is stored in accumulator. The carry bit is set to signify that the search was unsuccessful, abscissa is outside of the range.

Processor utilization

Function	Cycles max.	Execution Time max
table_look	87+n*19	(87+n*19)*50ns

Memory utilization

Function	ROM (words)	Stack levels	Registers used	RAM (words)
table_look	104+tables	none	3	7

```
         .bss    X_look,1,1
         .bss    temp,1,1
         .bss    X1,1,1
         .bss    X2,1,1
         .bss    Y1,1,1
         .bss    Y2,1,1
         .bss    remainder,1,1

size     .set    xxx        ;size of the array
iterations
         .set    xxx        ;number of iterations to complete the
                            ;search,
                            ;the size of this array is ;2^(xxx+1)
                            ;for example for a 2^9 valuestable,
                            ;xxx is ;equal 8

         .text
         •
         •
         •
         CALL    table_look
         •
         •
         •
table_look
         LDP     #temp
         SACL    X_look     ;load in Accu the searched abscissa
         LAR     AR0,#size  ;load in AR0, the size of array
         MAR     *,AR0
```

```
        MAR     *BR0+,AR3  ;half the size of the array
        LAR     AR3,#TableX
                           ;AR3 points to the beginning of the array
        LAR     AR4,#iterations
                           ;Number of iterations, table size is 2^n

        SAR     AR3,temp   ;Load Accu with address of the first
        LAC     temp       ;value in abscissa table
        TBLR    temp       ;Transfert the first abscissa in
                           ;temporary variable
        LAC     X_look
        SUB     temp       ;compare the searched abscissa with
                           ;pointed abscissa
        BCND    outside,LT ;error if the abscissa is smaller
                           ;than the first abscissa of the table

again
        BCND    found,EQ   ;if searched abscissa is equal pointed
abscissa
        BCND    inf,GT     ;if searched abscissa is greater
                           ; than pointed abscissa
        MAR     *0-,AR0    ;if too high on array, jump back
                           ;in the table
        B       end_sup
inf     MAR     *0+,AR0    ;if too low on array, jump
                           ;foward in the table
end_sup
        MAR     *BR0+,AR4  ;half the search part
        SAR     AR3,temp   ;Load Accu with address of
        LAC     temp       ;the new pointed abscissa
        TBLR    temp       ;transfert pointed abscissa in temporay
                           ;variable
        LAC     X_look
        SUB     temp       ;compare searched abscissa with pointed
                           ;abscissa
        BANZ    again,AR3  ;repeat iteration n times

nothere                    ;exact abscissa has not been found, an
        SAR     AR3,temp   ;interpolation has to be performed
        BCND    part_pos,GT;test if searched abscissa is greater
                           ;or smaller ;than pointed abscissa

                           ;if abscissa pointed is greater than
                           ;searched abscissa.
        LAC     temp       ;
        TBLR    X2         ;X2=min abscissa of the interval
        SUB     #1h
        TBLR    X1         ;X1=max abscissa of the interval

        SUB     #TableX    ;point to the Y table
        ADD     #TableY

        TBLR    Y1         ;Y1=ordinate of X1
        ADD     #1h
        TBLR    Y2         ;Y2=ordinate of X2
        B       interpolate
```

```
part_pos
                                ;if abscissa pointed is smaller than
                                ;searched abscissa.
        LAC     temp
        TBLR    X1              ;X1=min abscissa of the interval
        ADD     #1h
        TBLR    X2              ;X2=max abscissa of the interval

        SUB     #TableX         ;point to Y table
        ADD     #TableY

        TBLR    Y2              ;Y2=ordinate of X2
        SUB     #1h
        TBLR    Y1              ;Y1=ordinate of X1

                                ;interpolation
                                ;Y=Y1+(x_look-X1)/(X2-X1)*(Y2-Y1)
interpolate
        LAC     X2
        SUB     X1              ;calculate X2-X1
        BCND    outside,LT      ;error if x_look is geater
                                ;than last abscissa
        SACL    remainder

        LAC     #8000h
        ABS
        RPTK    15
        SUBC    remainder       ;calculate 1/(X2-X1) << 15
        SACL    temp
        LT      temp
        LAC     X_look
        SUB     X1              ;calculate x_look-X1
        SACL    temp
        MPY     temp            ;calculate ratio =(x_look-X1)/(X2-X1)<<15
        SPL     temp
        LT      temp

        LAC     Y2
        SUB     Y1              ;calculate Y2-Y1
        SACL    temp

        MPY     temp            ;calculate ratio*(Y2-Y1)
        PAC
        SFL
        SACH    temp
        LAC     Y1
        ADD     temp            ;calculate Y=Y1+(x_look-X1)/(X2-X1)
                                ;*(Y2-Y1)
        B       end_interp

outside
        ZAC                     ;Accu is zeroed
        SETC    C               ;set the carry to inform main program
                                ;that the search was unsuccessful
        B       end_interp

found                           ;exact abscissa has been found
```

```
        SAR     AR3,temp
        LAC     temp        ;point to good address in TableY
        SUB     #TableX
        ADD     #TableY
        TBLR    temp        ;Store Y(x_look) in temporary register
        LAC     temp        ;Y(x_look) in Accu

end_interp
        RET


TableX .word   X(0)        ;table of abscissa in program data space
       .word   X(1)
       .word   X(2)
       .word   X(3)
       .word   X(4)
       .word   X(5)
       .word   X(6)
       .word   X(7)
       •
       •

TableY .word   Y(10h)      ;table of ordinate in program data space
       .word   Y(20h)
       .word   Y(30h)
       .word   Y(40h)
       .word   Y(50h)
       .word   Y(60h)
       .word   Y(70h)
       .word   Y(80h)
       •
       •

       .end
```

This function could be easily modified if the size of the search table is not a power of 2.

An implementation of this function in a main program with an example is given in the annexe.

5. Annexe

5.1 Fixed Step Table example

```
**************************************************************
*File Name:     M_table.asm                                  *
*Project:       DMC Mathematical Library                     *
*Originator:    Pascal DORSTER (Texas Intruments)            *
*                                                            *
*Description:   Simple main which call a table Look-up       *
*               function with fixed step table               *
*                                                            *
*                                                            *
*Processor:     C2xx                                         *
*                                                            *
*Status:                                                     *
*                                                            *
*Last Update:   20 Sept 96                                   *
*____________________________________________________________*
*Date of Mod    |          DESCRIPTION                       *
*---------------|--------------------------------------------*
*               |                                            *
*               |                                            *
**************************************************************

.mmregs

.sect "vectors"
b        _c_int0
b        $

***************************************************
* Variable
***************************************************
.bss     Xdata,1,1
.bss     indice,1,1
.bss     remainder,1,1
.bss     Y1,1,1
.bss     Y2,1,1
.bss     temp,1,1


**************************************************
* Main routine
**************************************************
         .text

_c_int0:
         LAC    #18h
         CALL   Look_fixed_table
```

```
*************************************************************
*Routine Name:  look_fixed_table                            *
*Project:       DMC Mathematical Library                    *
*Originator:    Pascal DORSTER (Texas Intruments)           *
*                                                           *
*Description:   Look-up Table + Interpolation program for    *
*               C2xx fixed step table                       *
*               Assembly calling funtion                    *
*                                                           *
*Status:                                                    *
*                                                           *
*Processor:     C2xx                                        *
*                                                           *
*Calling convention:                                        *
*               Input : in Accu abscissa                    *
*               Output : Y(abscissa) in Accu                *
*                                                           *
*Last Update:   20 Sept 96                                  *
*                                                           *
*Date of Mod    |DESCRIPTION                                *
*---------------|-------------------------------------------*
*               |                                           *
*               |                                           *
*************************************************************


Look_fixed_table
  LDP    #Xdata             ;isolate the indice by a /8
  SACL   Xdata
  LAC    Xdata,16-3         ;integer position in the table
  SACH   indice             ;remainder
  SACL   remainder          ;
  LALK   tableY             ;address of beginning of the table
  ADD    indice             ;address of the nearest first indice
  SACL   indice             ;temporary
  SUB    #tableY_end
  BCND   outside,GT
  BCND   last,EQ
  LAC    remainder
  TBLR   Y1
  ADD    #1h
  TBLR   Y2                 ;Load Y2 with the ordonate of the
                            ;nearest second indice
  LAC    Y2
  SUB    Y1                 ;difference between the two Y value
  SACL   temp
  LT     remainder
  MPY    temp
  SPH    temp               ;interpolation between Y1 and Y2
  LAC    Y1
  ADD    temp               ;Y(Xdata) in ACCU
  B      end_interp
outside                     ;abscissa is out of range
  ZAC
```

```
  SETC   C
  B      end_interp
  last                    ;abscissa point to the last table
                          ;value
  LAC    remainder
  TBLR   Y1
  LAC    Y1
end_interp
  RET

***************************************************
* Table
***************************************************

tableY   .word  10        ;Y(0)  =10
         .word  40        ;Y(8)  =40
         .word  80        ;Y(16) =80
tableY_end
         .word  200       ;Y(24) =200
         .end
```

5.2 Generic Table example

```
*************************************************************
*File Name:     M_table.asm                                  *
*Project:       DMC Mathematical Library                     *
*Originator:    Pascal DORSTER (Texas Intruments)            *
*                                                            *
*Description:   Simple main which call a table Look-up       *
*               function not fixed step table                *
*                                                            *
*Processor:     C2xx                                         *
*                                                            *
*Status:                                                     *
*                                                            *
*Last Update:   20 Sept 96                                   *
*                                                            *
*Date of Mod    |          DESCRIPTION                       *
*---------------|--------------------------------------------*
*               |                                            *
*               |                                            *
************************************************************

        .mmregs

        .sect "vectors"
        b      _c_int0
        b      $

*************************************************************
* Variables                                                  *
*************************************************************
  .bss   X_look,1,1
  .bss   temp,1,1
  .bss   X1,1,1
  .bss   X2,1,1
  .bss   Y1,1,1
  .bss   Y2,1,1
  .bss   remainder,1,1

size     .set   08h        ;size of the array
iterations.set  2h         ;number of iterations to complete
                           ;the search,the size of this array
                           ;is 2^(xxx+1)
                           ;for example for a 2^9 values table,
                           :xxx is equal 8

        .text

*************************************************************
* Main                                                       *
*************************************************************
_c_int0:
```

```
         LAC    #32h
         CALL   table_look

*************************************************************
*Routine Name:  table_look                                  *
*Project:       DMC Mathematical Library                    *
*Originator:    Pascal DORSTER (Texas Intruments)           *
*                                                           *
*Description:   Look-up Table + Interpolation program       *
*               for C2xx                                    *
*               not fixed step table                        *
*               bit reversed table look-up                  *
*               Linear Interpolation                        *
*               Tables in Program Memory                    *
*               Size table is an integer power of 2         *
*               Boundary condition management: abscissa     *
*               out of range                                *
*               Assembly calling funtion                    *
*                                                           *
*Status:                                                    *
*                                                           *
*Processor:     C2xx                                        *
*                                                           *
*Calling convention:                                        *
*               Input : Abscissa in Accu                    *
*               Output: ordonates Y(x_look) in Accu         *
*               Carry bit is set if out of range            *
*                                                           *
*Last Update:   20 Sept 96                                  *
*___________________________________________________________*
* Date of Mod   |          DESCRIPTION                      *
*---------------|-------------------------------------------*
*               |                                           *
*               |                                           *
*************************************************************


table_look
  LDP    #temp
  SACL   X_look        ;load in Accu the searched abscissa
  LAR    AR0,#size     ;load in AR0, the size of array
  MAR    *,AR0
  MAR    *BR0+,AR3     ;half the size of the array
  LAR    AR3,#TableX   ;AR3 points to the beginning
                       ;of the array
  LAR    AR4,#iterations
                       ;Number of iterations,
                       ;table size is 2^n

  SAR    AR3,temp      ;Load Accu with address of
                       ;the first value in
  LAC    temp          ;abscissa table
  TBLR   temp          ;Transfer the first abscissa
                       :in temporary
```

```
                                ;variable
   LAC     X_look
   SUB     temp                 ;compare the searched abscissa with
                                ;pointed abscissa
   BCND    outside,LT           ;error if the abscissa is smaller
                                ;than the first abscissa of the table

again
   BCND    found,EQ             ;if searched abscissa is equal
                                ;pointed abscissa
   BCND    inf,GT               ;if searched abscissa is greater
                                ;than pointed abscissa
   MAR     *0-,AR0              ;if too high on array, jump back
                                ;in the table
   B       end_sup
inf
   MAR     *0+,AR0              ;if too low on array, jump
                                ;foward in the table
end_sup
   MAR     *BR0+,AR4            ;half the search part
   SAR     AR3,temp             ;Load Accu with address of
   LAC     temp                 ;the new pointed abscissa
   TBLR    temp                 ;transfer pointed abscissa in
                                ;temporary variable
   LAC     X_look
   SUB     temp                 ;compare searched abscissa with
                                ;pointed abscissa
   BANZ    again,AR3            ;repeat iteration n times

nothere                         ;exact abscissa has not been found, an
   AR      AR3,temp             ;interpolation has to be performed
   BCND    part_pos,GT          ;test if searched abscissa is greater
                                ;or smaller than pointed abscissa

                                ;if abscissa pointed is greater than
                                ;searched abscissa.
   LAC     temp                 ;
   TBLR    X2                   ;X2=min abscissa of the interval
   SUB     #1h
   TBLR    X1                   ;X1=max abscissa of the interval

   SUB     #TableX              ;point to the Y table
   ADD     #TableY

   TBLR    Y1                   ;Y1=ordinate of X1
   ADD     #1h
   TBLR    Y2                   ;Y2=ordinate of X2
   B       interpolate

part_pos
                                ;if abscissa pointed is smaller
                                ;than searched abscissa.
   LAC     temp
```

```
  TBLR   X1                ;X1=min abscissa of the interval
  ADD    #1h
  TBLR   X2                ;X2=max abscissa of the interval

  SUB    #TableX           ;point to Y table
  ADD    #TableY

  TBLR   Y2                ;Y2=ordinate of X2
  SUB    #1h
  TBLR   Y1                ;Y1=ordinate of X1

                           ;interpolation
                           ;Y=Y1+(x_look-X1)/(X2-X1)*(Y2-Y1)
interpolate
  LAC    X2
  SUB    X1                ;calculate X2-X1
  BCND   outside,LT        ;error if x_look is greater than
                           ;last abscissa
  SACL   remainder

  LAC    #8000h
  ABS
  RPTK   15
  SUBC   remainder         ;calculate 1/(X2-X1) << 15
  SACL   temp
  LT     temp
  LAC    X_look
  SUB    X1                ;calculate x_look-X1
  SACL   temp
  MPY    temp              ;calculate
                           ;ratio =(x_look-X1)/(X2-X1)<<15
  SPL    temp
  LT     temp

  LAC    Y2
  SUB    Y1                ;calculate Y2-Y1
  SACL   temp

  MPY    temp              ;calculate ratio*(Y2-Y1)
  PAC
  SFL
  SACH   temp
  LAC    Y1
  ADD    temp              ;calculate Y=Y1+(x_look-X1)/(X2-X1)
                           ;*(Y2-Y1)
  B      end_interp

outside
  ZAC                      ;Accu is zeroed
  SETC   C                 ;set the carry to inform main program
                           ;that the search was unsuccessful
  B      end_interp
```

```
found                       ;exact abscissa has been found
  SAR    AR3,temp
  LAC    temp               ;point to good address in TableY
  SUB    #TableX
  ADD    #TableY
  TBLR   temp               ;Store Y(x_look) in temporary register
  LAC    temp               ;Y(x_look) in Accu

end_interp
  RET

TableX   .word  10h         ;X(0)
         .word  30h         ;X(1)
         .word  35h         ;X(2)
         .word  50h         ;X(3)
         .word  60h         ;X(4)
         .word  65h         ;X(5)
         .word  70h         ;X(6)
         .word  90h         ;X(7)


TableY   .word  05h         ;Y(10h)
         .word  10h         ;Y(30h)
         .word  16h         ;Y(35h)
         .word  22h         ;Y(50h)
         .word  40h         ;Y(60h)
         .word  60h         ;Y(65h)
         .word  65h         ;Y(70h)
         .word  85h         ;Y(90h)

         .end
```