Tables ease microcontroller programming

Abel Raynus, Armatron International, Malden, MA

When creating microcontroller firmware, you often need to work with data arrays. Tables make easy work of data arrays, such as those for digital-code transformation, correction for sensor linearity, sophisticated calculations, and multiple output organization. **Table 1** shows how you can organize data in a table. Outputs A, B, and C have values based on the input value, V.

When using a lookup table, choose the proper microcontroller input and outputs. Assign values for input and outputs data in **Table 2**. These data can consist of constants in binary, hexadecimal, or decimal format or names. For names, you should assign a constant value to each one. For example:

```
data1  equ  $0a
data2  equ  $0b
data3  equ  $0c
data3  equ  $0d
```

Next, put the data from **Table 2** in either the beginning or the end of ROM, which makes the data easy to find. For definition of 1-byte data storage, use pseudo operators FCB or DB. For storage of data comprising 2 bytes, use FDB or DW, as in the following example:

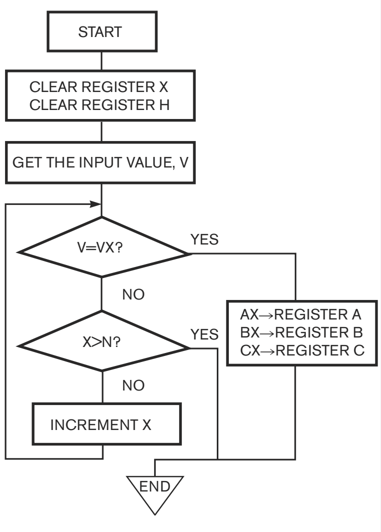

Figure 1 You can use a look-up table in microcontroller code.

```
ORG ROM
Vx  FCB  0T,2T,4T,6T
Ax  FCB  data1,data2,data3,data4
Bx  FCB  $aa,$bb,$cc,$dd
Cx  FDB  $1122,$3344,$5566,$7788
```

Note that commas separate the data. Don't place a comma after the last data, or it will be considered as $00.

When working with tables, you should always use indexed addressing mode. It provides access to data using variable addresses. Most microcontrollers have two index registers, X and H. Register X contains the low byte of the conditional address of the operand; H contains the high byte. The algorithm of working with tables is straightforward. After you detect the input value, you should then compare it with the table's input data. The X index determines this value, starting with X=0 and ending with X=N. In this example, N=4. When you find table data equal to the input value, you use the corresponding X as an index to load the output registers with their values. In the case of 2-byte numbers, you should load the output registers separately, first with a high byte and then with a low one. **Figure 1** illustrates this process.

The **listing** of assembler code is available from the online version of this Design Idea at www.edn.com/article/100422dib. In the **listing**, you can double-check the table content in memory at addresses $F800 through $F813. The listing uses Freescale (www.freescale.com) assembler because most of the appropriate applications employ inexpensive, 8-bit microcontrollers from Freescale's HC08 Nitron family. You can, however, use this approach with any type of microcontroller and assembly language.EDN

TABLE 1 OUTPUT VALUES VERSUS INPUT VALUES

Input V	Output A	Output B	Output C
V1	A1	B1	C1
V2	A2	B2	C2
. . . .	. . . .	. . . .	. . . .
VN	AN	BN	CN

TABLE 2 INPUT AND OUTPUT VALUES

Input V	Output A	Output B	Output C
V1=0T	data1	$aa	$1122
V2=2T	data2	$bb	$3344
V3=4T	data3	$cc	$5566
V4=6T	data4	$dd	$7788