Table-Driven Software

Data Tables

used to lookup values instead of computing them
generally trading off between speed and space

Code Tables

used to implement multi-way branches
more flexible than nested if/else
very similar to data tables, except the data is an address (and maybe more)

Data Table Examples

```
SqrtTable1:

        LDI     ZL, LOW(2 * SquareRoots1)   ;get the start of the table
        LDI     ZH, HIGH(2 * SquareRoots1)
        EOR     R0, R0                      ;zero R0 for carry propagation
        ADD     ZL, R16                     ;add in the table offset
        ADC     ZH, R0

        LPM     R0, Z                       ;and get the square root


EndSqrtTable1:                              ;have the square root - return
        RET


SquareRoots1:

        .DB      0,  1,  1,  2,  2,  2,  2,  3,  3,  3,  3,  3,  3,  4,  4,  4
        .DB      4,  4,  4,  4,  4,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  6
        .DB      6,  6,  6,  6,  6,  6,  6,  6,  6,  6,  6,  7,  7,  7,  7,  7
        .DB      7,  7,  7,  7,  7,  7,  7,  7,  7,  8,  8,  8,  8,  8,  8,  8
        .DB      8,  8,  8,  8,  8,  8,  8,  8,  8,  9,  9,  9,  9,  9,  9,  9
        .DB      9,  9,  9,  9,  9,  9,  9,  9,  9,  9,  9, 10, 10, 10, 10, 10
        .DB     10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 11
        .DB     11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11
        .DB     11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12
        .DB     12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 13, 13, 13
```

• • •

Data Table Examples

```
SqrtTable2:

SqrtTable2Init:                         ;get ready for the table lookup
        LDI     ZL, LOW(2 * SquareRoots2)   ;get the start of the table
        LDI     ZH, HIGH(2 * SquareRoots2)

SqrtLookupLoop:                         ;loop, looking for the passed value
        LPM     R0, Z+                  ;get next square root value from table
        CP      R0, R16                 ;compare passed value with table
        BRLO    SqrtLookupLoop          ;keep trying if haven't reached passed
        ;BRSH   FoundSqrt               ;   value yet, otherwise found it
                                        ;have to exit loop because last entry
                                        ;   is the max value for R16
FoundSqrt:                              ;found the square root, it's the table offset
        LDI     R16, LOW(2 * SquareRoots2) + 1   ;get start of table to find offset
        MOV     R0, R16                         ;   +1 because of post-inc in LPM
        SUB     R0, ZL                  ;compute the -offset (works even if
        NEG     R0                      ;   cross pages), so negate
        ;RJMP   EndSqrtTable2           ;and all done

EndSqrtTable2:                          ;have the square root - return
        RET


SquareRoots2:

        .DB       0,   2,   6,  12,  20,  30,  42,  56
        .DB      72,  90, 110, 132, 156, 182, 210, 240, 255, 255
```

Data Table Examples

```
  TestSwitchLEDs:                           ;do the DisplaySwitchLEDs tests
          LDI     ZL, LOW(2 * TestSWTab)    ;start at the beginning of the
          LDI     ZH, HIGH(2 * TestSWTab)   ;   DisplaySwitchLEDs test table

  TestSwitchLEDsLoop:
          LPM     R16, Z+                   ;get the DisplaySwitchLEDs arguments
          LPM     R17, Z+                   ;   from the table
          RCALL   DisplaySwitchLEDs         ;call the function

          LPM     R16, Z                    ;get the time delay from the table
          RCALL   Delay16                   ;and do the delay

          ADIW    Z, 1                      ;skip the padding byte

          LDI     R20, HIGH(2 * EndTestSWTab)  ;setup for end check
          CPI     ZL, LOW(2 * EndTestSWTab) ;check if at end of table
          CPC     ZH, R20
          BRNE    TestSwitchLEDsLoop        ;and keep looping if not done
          ;BREQ   TestDisplayHex            ;otherwise test DisplayHex function

   TestSWTab:
                  ;Argument (low /high)     Delay (10 ms)    Padding
           .DB     0b11111111, 0b11111111, 100,             0    ;all on
           .DB     0b00000000, 0b00000000, 100,             0    ;all off
           .DB     0b10011101, 0b10000111, 100,             0
           .DB     0b01100010, 0b11101000, 100,             0
           .DB     0b00010010, 0b00010100, 100,             0
```

Multi-Way Branch Example

```
MultiWayBranch:
        LDI     ZL, LOW(2 * LookupTable);get the start of the table
        LDI     ZH, HIGH(2 * LookupTable)
        LDI     R17, TABLE_ENTRIES      ;and the number of entries

LookupLoop:                             ;loop, looking for the passed value
        LPM     R18, Z+                 ;get lookup value from table
        CP      R18, R16                ;is this the matching value
        BREQ    HaveMatch               ;if so, take care of it
        ;BRNE   NotMatching             ;otherwise not a match yet

NotMatching:                            ;doesn't match entry yet
        ADIW    Z, TABLE_ENTRY_SIZE - 1 ;move to next entry (already did +1)
        DEC     R17                     ;update number of entries left
        BRNE    LookupLoop              ;and loop while still have entries
        ;BREQ   HaveMatch               ;otherwise no matching value in table
                                        ;   process default entry

HaveMatch:                              ;have entry we want
        LPM     R18, Z+                 ;get the address
        LPM     R19, Z
        MOVW    Z, R18                  ;move address into Z
        IJMP                            ;and jump to the address
```

```
; LookupTable
;
; Description:      This table contains pairs of key values and the associated
;                   processing labels.  There is also a padding byte to keep
;                   it on a word boundary
;
; Author:           Glen George
; Last Modified:    May 26, 2022

LookupTable:
                ;Key Value          Processing Lable              Padding
        .DB     KEY_1,      LOW(ProcessKey1), HIGH(ProcessKey1), 0

        .EQU    TABLE_ENTRY_SIZE = 2 * (PC - LookupTable)

        .DB     KEY_2,      LOW(ProcessKey2), HIGH(ProcessKey2), 0
        .DB     KEY_3,      LOW(ProcessKey3), HIGH(ProcessKey3), 0
        .DB     KEY_4,      LOW(ProcessKey4), HIGH(ProcessKey4), 0

        .EQU    TABLE_ENTRIES = (PC - LookupTable) / (TABLE_ENTRY_SIZE / 2)

        .DB     HIGH(DefaultKey), LOW(DefaultKey)       ;default processing
```

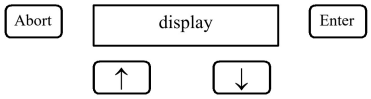
Command Table

Key Value	Action	Argument	Display
KEY_11	MoveForward	0	ForwArd
KEY_12	MoveReverse	0	rEvErSE
KEY_13	TurnLeft	0	LEFt
KEY_21	SetBaud	300	bAUd300
KEY_22	SetBaud	1200	bAUd1200
KEY_23	SetBaud	9600	bAUd9600
KEY_31	SetParity	PAR_EVEN	PArEvEn
KEY_32	SetParity	PAR_ODD	PArOdd
KEY_33	SetParity	PAR_MARK	PAr1
. . .	. . .	. . .	. . .

Assembly Table

```
CmdTable:
          ;Keycode        Function to Call              Arg  Display String
     .DB   KEY_11, LOW(MoveForward), HIGH(MoveForward),  0, "ForwArd "
     .DB   KEY_12, LOW(MoveReverse), HIGH(MoveReverse),  0, "rEvErSE "
     .DB   KEY_13, LOW(TurnLeft),    HIGH(TurnLeft),     0, "LEFt    "
     .DB   KEY_21, LOW(SetBaud),     HIGH(SetBaud),      3, "bAUd300 "
     .DB   KEY_22, LOW(SetBaud),     HIGH(SetBaud),     12, "bAUd1200"
     .DB   KEY_23, LOW(SetBaud),     HIGH(SetBaud),     96, "bAUd9600"
```

Menu System



Main Menu Table

Action	Arg	Display	Next Menu
MoveForward	0	FowArd	MainMenu
MoveReverse	0	rEvErSE	MainMenu
TurnLeft	0	LEFt	MainMenu
TurnRight	0	RIgHt	MainMenu
doNOP	0	SEt bAUd	BaudMenu
doNOP	0	SEt PAri	ParityMenu

Baud Menu Table

Action	Arg	Display	Next Menu
SetBaud	300	300	MainMenu
SetBaud	1200	1200	MainMenu
SetBaud	4800	4800	MainMenu
SetBaud	9600	9600	MainMenu
SetBaud	19200	19200	MainMenu
SetBaud	57600	57600	MainMenu

Assembly Tables

```
MainMenu:
        ;Initialization    Table      TableSize
    .DW   StartFirst,       MainTable, MAIN_SIZE

MainTable:
        ;Action        Arg   NextTable  Display String
    .DW   MoveFoward,     0,  MainMenu
    .DB   "FowArd  "
    .DW   MoveReverse,    0,  MainMenu
    .DB   "rEvErSE "
    .DW   TurnLeft,       0,  MainMenu
    .DB   "LEFt    "
    .DW   TurnRight,      0,  MainMenu
    .DB   "RIgHt   "
    .DW   doNOP,          0,  BaudMenu
    .DB   "Set bAUd"
    .DW   doNOP,          0,  ParityMenu
    .DB   "Set Pari"
    .EQU   MAIN_SIZE = (PC - MainTable) / 14

BaudMenu:
        ;Initialization    Table      TableSize
    .DW   GetBaud,          BaudTable, BAUD_SIZE

BaudTable:
        ;Action        Arg   NextTable  Display String
    .DW   SetBaud,      300,  MainMenu
    .DB   "300     "
    .DW   SetBaud,     1200,  MainMenu
    .DB   "1200    "
    .DW   SetBaud),    9600,  MainMenu
    .DB   "4800    "
    .EQU  BAUD_SIZE = (PC - BaudTable) / 14
```

SDCardInit

Command	Argument	Response
0	--	--
8	0x000001AA	0x00 → V2 0x04 → V1
		Version 1 Card
58	--	0x00FC0000 → OK 0x00031000 → Bad
55	--	0b0000000x → OK 0x7C → Bad
41	0x00000000	0x00 → Done 0x01 → Loop 0x7C → Bad
		Version 2 Card
--	--	0x000000AA → OK otherwise Bad
58	--	0x00FC0000 → OK 0x00031000 → Bad
55	--	0b0000000x → OK 0x7C → Bad
41	0x40000000	0x00 → Done 0x01 → Loop 0x7C → Bad
58	--	0x00000000 → OK (SD) 0x40000000 → OK (SDHC)

SDCardInit

Step	Cmd	Argument	Response			Non-Match		Match	
			Len	Mask	Compare	Error	Next Step	Error	Next Step
0	0	0x0000000095	1	0x00000000	0x00000000	F	1	F	1
1	8	0x000001AAFF	4	0x04000000	0x04000000	F	6	F	2
2	58	0x00000000FF	4	0x00FC0000	0x00000000	F	3	T	-4
3	-1	0x00000000FF	0	0x04000000	0x04000000	F	4	T	-4
4	55	0x00000000FF	1	0x7C000000	0x00000000	T	-4	F	5
5	41	0x00000000FF	1	0x01000000	0x01000000	F	-1	F	4
6	-1	0x00000000FF	0	0x000000FF	0x000000AA	T	-4	F	7
7	58	0x00000000FF	4	0x00FC0000	0x00000000	F	8	T	-4
8	55	0x00000000FF	1	0x7C000000	0x00000000	T	-4	F	9
9	41	0x40000000FF	1	0x01000000	0x01000000	F	10	F	8
10	58	0x00000000FF	4	0x40000000	0x40000000	F	-2	F	-3

-1 Version 1.x Card Ready

-2 Version 2.x SD Card Ready

-3 Version 2.x SDHC Card Ready

-4 Error Non-Compatible Card