AVR Freaks

DESIGN NOTE #014

KEYWORDS:
AT90S1200, Flash Table, EEPROM Table, LPM, LDI

Flash Tables for AVRs without the LPM Instruction

Introduction

This document describes an example of how to support look-up tables in an AT90S1200 (or on any other AVR) without using the LPM instruction.

The recommended solution is to put look-up tables in EEPROM. However, if the size of the EEPROM is inadequate, the program memory could be used in addition to this.

Code Example

The only instruction to move a byte from program memory to a register in the AT90S1200 is the LDI instruction. The LDI instruction will write new contents into a specified target register. Target registers are limited to the range R16 - R31.

When implementing Flash tables using the LDI instruction, a number of test and branch instruction are requried. It is important to limit the number of “supporting” branch and test instructions since these will add to the total code size. A code-efficient way of storing bytes in Flash memory is to place many consecutive LDI instructions in the program memory. This will move a block of data into a data buffer in the register file. This buffer can then be indexed using the “LD Rd, Z” instruction. The achievable block size is depending on how many bytes that can be spared from the register file during this operation. In the example give below only four bytes from the register file are used for the data block/data buffer.

Parts of the look-up table are still be placed in the EEPROM. In the enclosed code example four bytes of the table is placed in EEPROM and eight bytes in Flash. This is done to illustrate the possibility to have a table that is “larger” than the EEPROM. When a specific table address is looked-up in the code example given, the look-up function determines whether to look-up the data in EEPROM or Flash atomatically. In other words, the look-up table is a combination of EEPROM and Flash.

Code

```
;***************************************************************************
;** File name: LDI_flash_table.asm
;** The AT90S1200 is not equipped with an 'lpm' instuction
;** This code implements a 12 byte look-up table using
;** the EEPROM to store 4 bytes, and 'ldi' instructions
;** for the remaining 8 bytes.
;***************************************************************************
.include "1200def.inc"
.include "Const_table.asm"

.def     index        = r16     ;look up table address
.def     temp         = r17     ;general purpose temporary register

.equ     buffer_size  = 0x04    ;Number of bytes in register file buffer
.equ     buffer_start = 20      ;offset from r0 to first buffer register
.equ     table_size   = 12;     ;Total table size
.equ     e2_table_size= 4;      ;Size of the table located in eeprom

;***************************************************************************
;** Lookup table definitions
.def     table_result = r20     ;return register, usually 'buffer0'
.def     buffer0      = r20     ;Register file buffers
.def     buffer1      = r21     ;NOTE 1: Registers R16-R31 ONLY!
.def     buffer2      = r22     ;NOTE 2: Registers must be consecutive
.def     buffer3      = r23

;***************************************************************************
;** Interrupt Vector Table
.cseg
.org 0x00
  rjmp    reset

;***************************************************************************
;** Reset routine
;   - Reset the lookup table address (index)
;   - Set up port B as output for the result
reset:
  ldi     temp, 0xff
  out     DDRB, temp             ;Port B all outputs
  clr     index                  ;Reset index (the lookup table address)
  clr     r31                    ;Reset ZH

  rjmp    main

;***************************************************************************
;** Main loop
;   - decrement lookup table address (index)
;   - read lookup table
```

```
;   - copy lookup table contents to PORT B output
main:
  mov     ZL, index             ;store lookup address in ZL
  rcall   lookup_table_read
  out     PORTB, table_result
  inc     index                 ;increment the lookup table address
  cpi     index, table_size     ;If index out of table bounce
  brsh    end                   ; jump end
  rjmp    main                  ;else repeat main loop

;***********************************************************************
;** Table read
;   - Table address read from ZL register (R30)
;   - Return table contents in 'table_result' register
;   - buffer size can be altered
lookup_table_read:
  cpi     ZL, e2_table_size     ;if table address points to flase
  brsh    lookup_table_flash1   ; jump flash lookup routines

lookup_table_eeread:
  sbic    EECR, EEWE            ;wait for EEPROM write to end
  rjmp    lookup_table_eeread   ;repeat wait until EEPROM ready

  out     EEAR, ZL              ;write index to EEPROM address register
  sbi     EECR, EERE            ;issue EEPROM read command
  in      table_result, EEDR    ;read table contents to table_result register
  ret                           ;return to calling function

lookup_table_flash1:
  subi    ZL, e2_table_size+buffer_size
                                ;if index < e2_table_size + buffer_size
  brsh  lookup_table_flash2     ; jump to lookup_table_2
  ldi     buffer0, table04
  ldi     buffer1, table05
  ldi     buffer2, table06
  ldi     buffer3, table07
  rjmp    lookup_return_value   ;jump to return value routine

lookup_table_flash2:
  subi    ZL, buffer_size       ;Prepare Z, no jump required
  ldi     buffer0, table08
  ldi     buffer1, table09
  ldi     buffer2, table0a
  ldi     buffer3, table0b

lookup_return_value:
  subi    ZL, -(buffer_size + buffer_start)
                                ;add buffer offset
```

```
  ld      table_result, Z        ;read from buffer to 'table_result'
  ret                            ;return to calling function

end:
  rjmp    end
```

And the included file "Const_table.asm" is as follows:

```
;***********************************************************
;** file name: Const_table.asm
;** This file contains the constant tables, of which
;** 4 bytes are located in EEPROM and 8 bytes in flash
;** memory

;**EEPROM momory     contents
.eseg
.org 0x00
table00:  .db         0x00
table01:  .db         0x01
table02:  .db         0x02
table03:  .db         0x03

.cseg
;**Flash  memory     "contents"
.equ      table04     = 0x04
.equ      table05     = 0x05
.equ      table06     = 0x06
.equ      table07     = 0x07
.equ      table08     = 0x08
.equ      table09     = 0x09
.equ      table0a     = 0x0a
.equ      table0b     = 0x0b
.equ      table0c     = 0x0c
```