AVR102: Block Copy Routines

Features

- **Program Memory (Flash) to SRAM Copy Routine**
- **SRAM to SRAM Copy Routine**
- **Extremely Code Efficient Routines**
  Flash → SRAM: 6 Words,
  SRAM → SRAM: 5 Words
- **Runable Test/Example Program**

Introduction

This application note contains routines for transfer of data blocks.

- Flash to SRAM Copy. Copies data blocks of sizes up to 256 bytes from any Flash address to any SRAM address.
- SRAM to SRAM copy. Copies data blocks of sizes up to 256 bytes from any SRAM source address to any SRAM destination address.

The routines are described in detail in the following sections.

Flash to SRAM Copy - Subroutine “flash2ram”

Before calling this routine, the following input parameters must be set up by the user:

- “flashsize” - Register variable holding the size of the data block
- Z-pointer - 2 x source address of first Flash block byte
- Y-pointer - Destination address of first SRAM block byte

The routine uses the “LPM”-instruction which is the one to use for data transfer from Flash to Register File. The instruction selects the Flash word (16-bit) pointed to by the 15 most significant bytes of the Z-pointer. If LSB of the Z-pointer is 0, the lower byte of the Flash word is returned, otherwise the higher byte is returned. The returned byte is always found in R0. To obtain the 15 MSBs of Z with the correct word value, the user must be careful to load Z with 2 times the address of the first double byte in the table. The program listing shows how to handle this precaution.

During data transfer, the auto-increment facility of the pointers is utilized. Writing to SRAM is done using the “ST Y+,Rs”-instruction. The “LPM”-instruction does not support auto-increment or decrement. The “ADIW” instruction is used to increment the 16-bit pointer.

The register variable “flashsize” is used as a loop counter in the routine.

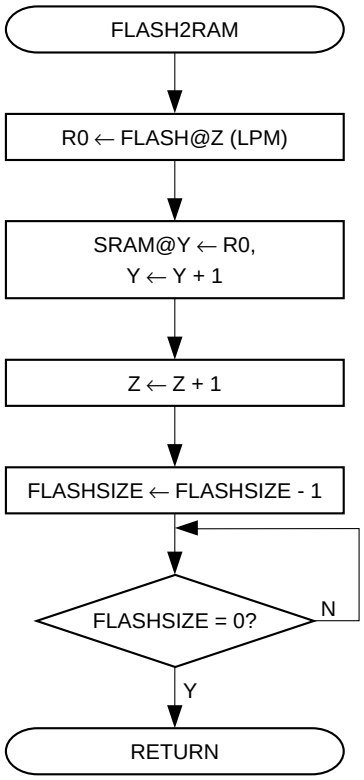

Figure 1. “flash2ram” Flow Chart

8-Bit AVR Microcontroller

Application Note

0933A-A–8/97

Table 1. “flash2ram” Register Usage

Register	Input	Internal	Output
R0		Temporary data storage	
R16	“flashsize” - size of data block to copy		
R28	“YL” - low address of SRAM data start		
R29	“YH” - high address of SRAM data start		
R30	“ZL” - low address of Flash data start		
R31	“ZH” - high address of Flash data start		

Table 2. “flash2ram” Performance Figures

Parameter	Value
Code Size (Words)	5 + return
Execution Time	10 x (block size) + return
Register Usage	• Low registers :1 • High registers :1 • Pointers :Y, Z
Interrupts Usage	None
Peripherals Usage	None

SRAM to SRAM Copy - Subroutine “ram2ram”

Before calling this routine, the following input parameters must be set up by the user:

- “ramsize” - Register variable holding the size of the data block
- Z-pointer - source address of first byte in block
- Y-pointer - destination address of first byte in block

By utilizing the pointer auto-increment facility, this routine is extremely compact. Moving one byte of data and incrementing both pointers requires 2 instructions only. Two more instructions are needed to decrement an check the register variable “ramsize” which is used as a loop counter in the routine. The data are temporarily stored in the register variable “ramtemp” during transfer.

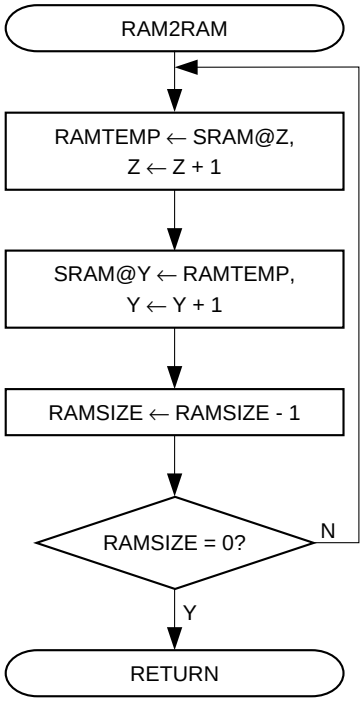

Figure 2. “ram2ram” Flow Chart

Table 3. “ram2ram” Register Usage

Register	Input	Internal	Output
R1		“ramtemp” - Temporary data storage	
R16	“ramsize” - size of data block to copy		
R28	“YL” - destination address low byte		
R29	“YH” - destination address high byte		
R30	“ZL” - source address low byte		
R31	“ZH” - source address high byte		

Table 4. “flash2ram” Performance Figures

Parameter	Value
Code Size (Words)	4 + return
Execution Time	6 x (block size) + return
Register Usage	• Low registers :1 • High registers :1 • Pointers :Y, Z
Interrupts Usage	None
Peripheral Usage	None

Test/Example Program

The application note program file contains a runable test program which uses the copy application routines to copy 20 bytes of data from Flash to SRAM, then make a second copy of the data at another SRAM location. The test program is made for use with an AT90S8515. By changing including another “*def.inc” file, the program can be used with any other AVR MCU with SRAM.