

Claus Kuhnel

BASCOM
Programming of
Microcontrollers with Ease

An Introduction
by Program Examples

BASCOM Programming of Microcontrollers with Ease:
An Introduction by Program Examples

Copyright © 2001 Claus Kuhnel

All rights reserved. No part of this work may be reproduced in any
form except by written permission of the author.

All rights of translation reserved.

Publisher and author assume no responsibility for any errors that
may arise from the use of devices and software described in this
book.

Universal Publishers/uPUBLISH.com
USA • 2001

ISBN: 1-58112-671-9

www.upublish.com/books/kuhnel.htm

Preface

The microcontroller market knows some well introduced 8-bit microcontroller families like Intel's 8051 with its many derivatives from different manufacturers, Motorola's 6805 and 68HC11, Microchip's PICmicros and Atmel's AVR.

The 8051 microcontroller family has been well-known over many years. The development of new derivatives is not finished yet. From time to time new powerful derivatives are announced.

You will find derivatives from Philips, Dallas, Analog Devices and Cygnal and others with the known 8051 core but enhanced clock and peripherals. For example, complete analog-to-digital and digital-to-analog subsystems were integrated in some chips.

Atmel developed the AVR microcontroller family which is well suited for high-level language programming and in-system programming.

For all those microcontrollers there is development software ranging from simple assemblers for DOS to integrated development environments for Windows95/98/NT on the market.

Apart from programming environments as they are offered, for example, by KEIL, IAR or E-LAB Computer for professional applications, also the more economical and nonetheless sufficiently equipped development environments can maintain ground.

BASCOM-8051 and BASCOM-AVR are development environments built around a powerful BASIC compiler which is suited for project handling and program development for the 8051 family and its derivatives as well as for the AVR microcontrollers from Atmel.

The programming of microcontrollers using BASCOM-8051 (version 2.0.4.0) and BASCOM-AVR (version 1.11.3.0) will be described in this book.

Some applications help understand the usage of BASCOM-8051 and BASCOM-AVR.

Acknowledgement

I should like to thank the following:

- in the first place, Mark Alberts of MCS Electronics, who developed the BASCOM programming environment at an outstanding price-performance ratio,
- Atmel for the development of the AVR RISC microcontrollers which introduced new capabilities into the microcontroller families,
- Christer Johansson of High Tech Horizon, who supports safe communication of microcontrollers and PC by the development and free distribution of the S.N.A.P. protocol and the necessary tools effectively and
- Lars Wictorsson of LAWICEL for the development of the CANDIPs, microcontroller modules with CAN interface.

Contents

1 Supported Microcontrollers	9
1.1 8051 Family	9
1.2 AVR Family	11
2 BASCOM.....	23
2.1 BASCOM Demos.....	23
2.2 BASCOM Commercial Versions.....	25
2.3 Update of BASCOM Commercial Versions	25
2.4 BASCOM Projects	27
2.4.1 Working on Projects	27
2.4.2 BASCOM Options	28
2.5 BASCOM Tools	37
2.5.1 Simulation	37
2.5.2 Terminal Emulator	40
2.5.3 LCD Designer	42
2.5.4 Library Manager	46
2.5.5 Programming Devices	50
2.6 Hardware for AVR RISC Microcontroller	55
2.6.1 DT006 AVR Development Board	55
2.6.2 AVR-ALPHA with AT90S2313	56
2.7 Instead of "Hello World"	57
2.7.1 AVR.....	57
2.7.2 8051	58
2.7.3 Things in Common.....	59
2.7.4 Simulation	64
2.8 BASCOM Help System.....	67
3 Some BASCOM Internals	69
3.1 Building new instructions	69

3.2 Parameters for Subroutines in BASCOM-AVR.....	71
3.3 BASIC & Assembler.....	73
3.3.1 AVR.....	74
3.3.2 8051.....	75
4 Applications.....	77
4.1 Programmable Logic.....	77
4.2 Timer and Counter.....	81
4.2.1 AVR.....	81
4.2.2 8051.....	104
4.3 LED Control.....	107
4.3.1 Single LED.....	107
4.3.2 Seven-Segment Displays.....	108
4.3.3 Dot-Matrix Displays.....	114
4.4 LCD Control.....	119
4.4.1 Direct Control.....	119
4.4.2 LCD with Serial Interface.....	122
4.5 Connecting Keys and Keyboards.....	128
4.5.1 Single Keys.....	129
4.5.2 Matrix Keypad.....	132
4.5.3 PC-AT Keyboard.....	136
4.6 Data Input by IR Remote Control.....	140
4.7 Asynchronous Serial Communication.....	143
4.8 1-WIRE Interface.....	151
4.9 SPI Interface.....	161
4.10 I ² C Bus.....	167
4.11 Scalable Network Protocol S.N.A.P.....	173
4.11.1 S.N.A.P. Features.....	174
4.11.2 Description of S.N.A.P. Protocol.....	175
4.11.3 S.N.A.P. Monitor.....	179
4.11.4 Digital I/O.....	183

4.12 CANDIP - Interface to CAN	197
4.13 Random Numbers	209
4.14 Moving Average	214
5 Appendix	219
5.1 Decimal-Hex-ASCII Converter.....	219
5.2 DT006 Circuit Diagram	220
5.3 Characters in Seven-Segment Display.....	222
5.4 BASIC Stamp II	223
5.5 Literature	224
5.6 Links	225
6 Index	231

1 Supported Microcontrollers

BASCOM is an Integrated Development Environment (IDE) that supports the 8051 family of microcontrollers and some derivatives as well as Atmel's AVR microcontrollers. Two products are available for the various microcontrollers - BASCOM-8051 and BASCOM-AVR.

In a microcontroller project one needs to know the hardware base, i.e. the microcontroller with internal and connected peripherals, and the software used, i.e. IDE handling, programming and debugging.

In this first chapter, let's have a look at the supported microcontrollers. A general overview will be given only; the various parts are documented by the manufacturers in more detail. You may also search the web for more information and documentation on all the microcontrollers dealt with here.

1.1 8051 Family

The 8051 is an accumulator-based microcontroller featuring 255 instructions. A basic instruction cycle takes 12 clocks; however, some manufacturers redesigned the instruction-execution circuitry to reduce the instruction cycle.

The CPU has four banks of eight 8-bit registers in on-chip RAM for context switching. These registers reside within the 8051's lower 128 bytes of RAM along with a bit-operation area and scratchpad RAM. These lower bytes can be addressed directly or indirectly by using an 8-bit value. The upper 128 bytes of on-chip data RAM encompass two overlapping address spaces. One space is for directly addressed special-function registers (SFRs); the other space is for indirectly addressed RAM or stack. The SFRs define peripheral operations and configurations. The 8051 also has 16 bit-addressable bytes of on-chip RAM for flags or variables.

Without external circuitry, the maximum address range of all 8051 processors is 64 Kbytes of program memory and 64 Kbytes of data memory. External means can be made use of to increase this address space.

Register indirection uses an 8-bit register for an on-chip RAM address; an off-chip address requires an 8- or 16-bit data-pointer register (DPTR). The original 8051 has only one DPTR. Derivatives from Atmel, Dallas, and Philips have two DPTRs. Siemens microcontrol-

lers have eight DPTRs. The 8051 microcontroller has bidirectional and individually addressable I/O lines.

The 8051 performs extensive bit manipulation via instructions, such as set, clear, complement, and jump on bit set or jump on bit clear, only for a 16-byte area of RAM and some SFRs. It can also handle AND or OR bits with a carry bit. The Dallas versions have variable-length move-external-data instructions. Math functions include add, subtract, increment, decrement, multiply, divide, complement, rotate, and swap nibbles. Some of the Siemens devices have a hardware multiplier/divider for 16-bit multiply and 32-bit divide. Figure 1 shows the block diagram of an 8051 [1].

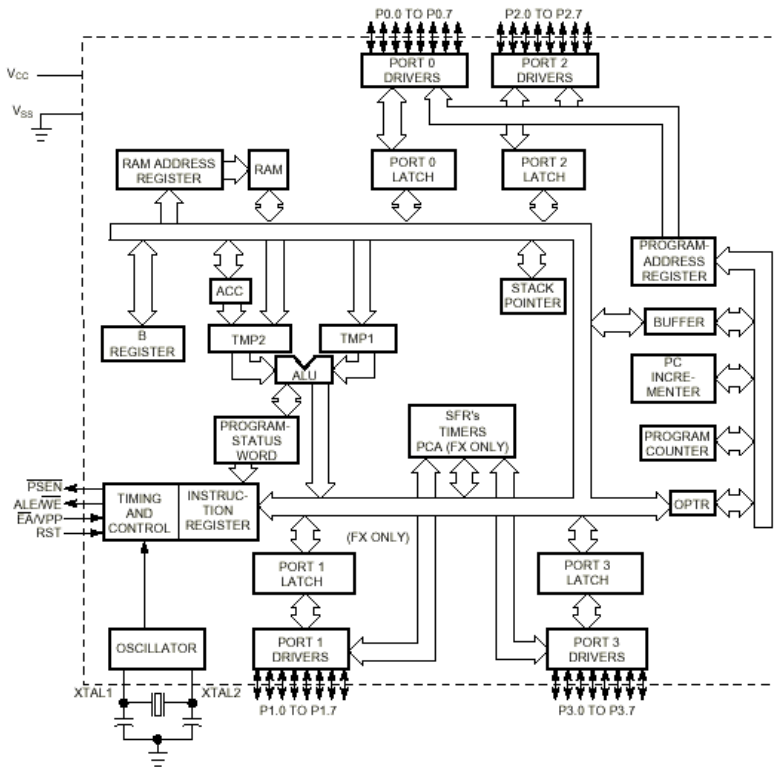


Figure 1 Block diagram 8051

To elucidate the differences in the derivatives, Figure 2 shows the block diagram of the C8051F0000 microcontroller from Cygnal [2].

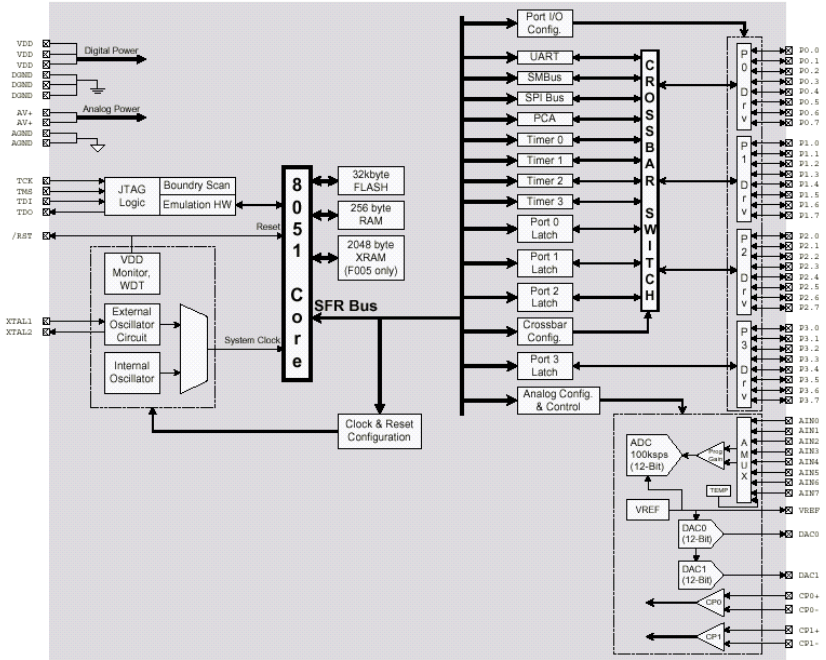


Figure 2 Block diagram C8051F0000

This is not the place to discuss the hardware aspects of the different derivatives of the 8051 family. The examples are meant to show that not all parts named 8051 are alike; the core is the same but the internal peripherals differ significantly.

Once you know the used hardware, you can organize the access to the resources of the chosen microcontroller.

1.2 AVR Family

Since Atmel's AVR microcontrollers were introduced to the market only a few years ago, they are not so well known as the 8051 controllers. Therefore, this interesting microcontroller family should be described in more detail.

Atmel's AVR microcontrollers use a new RISC architecture which has been developed to take advantage of the semiconductor integration and software capabilities of the 1990's. The resulting microcontrollers offer the highest MIPS/mW capability available in the 8-bit microcontrollers market today.

The architecture of the AVR microcontrollers was designed together with C-language experts to ensure that the hardware and software work hand-in-hand to develop a highly efficient, high-performance code.

To optimize the code size, performance and power consumption, AVR microcontrollers have big register files and fast one-cycle instructions.

The family of AVR microcontrollers includes differently equipped controllers - from a simple 8-pin microcontroller up to a high-end microcontroller with a large internal memory. The Harvard architecture addresses memories up to 8 MB directly. The register file is "dual mapped" and can be addressed as part of the on-chip SRAM, whereby fast context switches are possible.

All AVR microcontrollers are based on Atmel's low-power nonvolatile CMOS technology. The on-chip in-system programmable (ISP), downloadable flash memory permits devices on the user's circuit board to be reprogrammed via SPI or with the help of a conventional programming device.

By combining the efficient architecture with the downloadable flash memory on the same chip, the AVR microcontrollers represent an efficient approach to applications in the "Embedded Controller" market.

Table 1 shows an overview of the devices available today, including the configuration of the internal memory and I/O. Further information can be found on Atmel's web site [<http://www.atmel.com>] and in the literature [3].

<i>Device</i>	<i>Flash [KB]</i>	<i>EEPROM</i>	<i>SRAM</i>	<i>I/O Pins</i>
ATtiny11	1	0	0	6
ATtiny12	1	64	0	6
ATtiny22	2	128	90	5
AT90S1200	1	64	0	15
AT90S2313	2	128	128	15
AT90S2323	2	128	128	3
AT90S2343	2	128	128	5
AT90S2333	2	128	128	20
AT90S4414	4	256	256	32
AT90S4433	4	256	128	20
AT90S4434	4	256	256	32
AT90S8515	8	512	512	32
AT90S8534	8	512	256	15
AT90S8535	8	512	512	32
ATmega603	64	2K	4K	48
ATmega103	128	4K	4K	48

Table 1 AVR microcontrollers and their resources

The internal resources of the AVR microcontrollers will be considered with AT90S8515 used as an example. Figure 3 shows the block diagram of an AT90S8515.

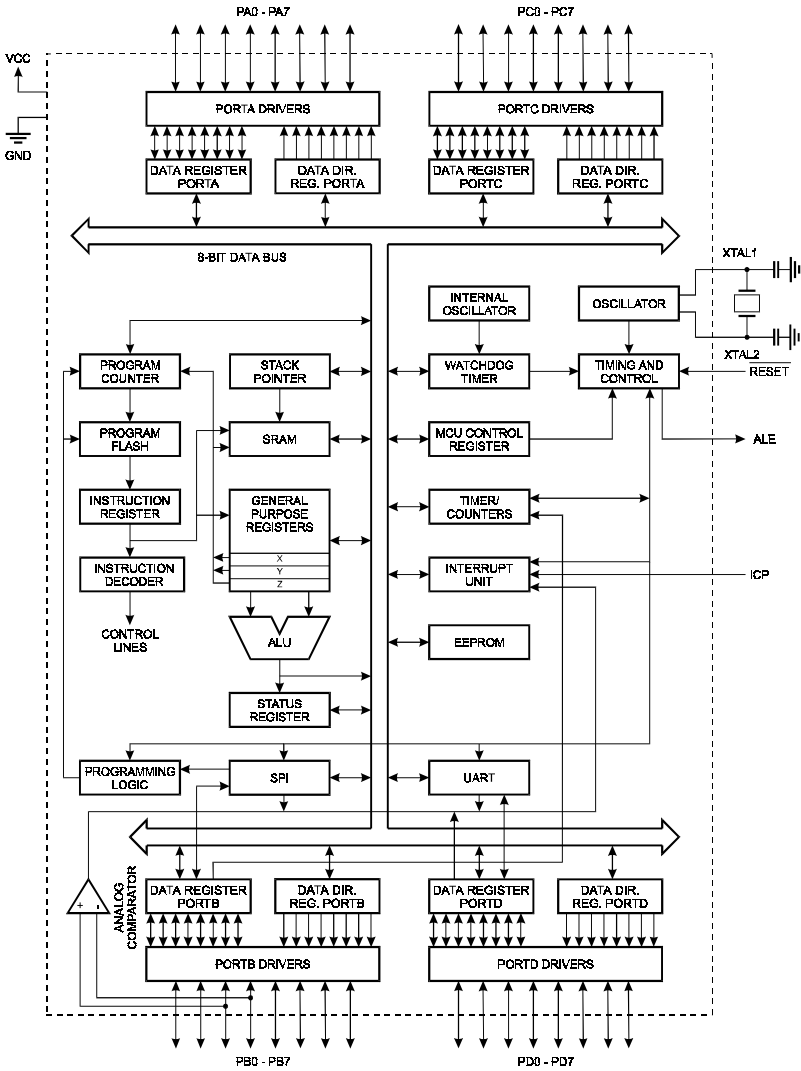


Figure 3 Block diagram AT90S8515

The I/O storage area covers 64 addresses for the peripheral device functions of the CPU, like control registers, Timer/Counter and other I/O functions. Figure 4 shows memory maps of the AT90S8515 program and data memory.

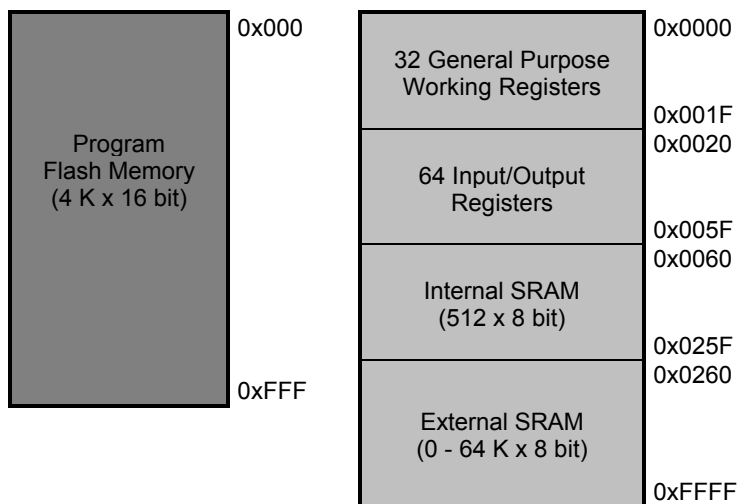


Figure 4
Memory maps for program and data memory for AT90S8515

The AVR microcontrollers make use of a Harvard structure with separate memories and busses for programs and data

A flexible interrupt module has its control register in the I/O memory area, too. All interrupts have separate interrupt vectors in an interrupt vector table at the beginning of the program memory. The priority level of each interrupt vector is dependent on its position in the interrupt vector table. The higher the priority of a respective interrupt, the lower is the address of the interrupt vector. All interrupts are maskable and can be enabled or disabled by a Global Interrupt Enable/Disable.

To get an impression of the available peripheral functions, the peripheral functions of the AT90S8515 will be listed here in brief as an example.

Timer/Counter

One 8-bit and one 16-bit Timer/Counter are available in conjunction with a flexible 10-bit prescaler for different timer and counter applications.

Both Timer/Counter units can operate independently as a timer with internal clock or as a counter with external triggering. The prescaler divides the internal clock into four selectable timer clocks (CK/8, CK/64, CK/256 and CK/1024).

The 8-bit Timer/Counter is a simple UpCounter.

The 16-bit Timer/Counter is more complex and supports two Output Compare functions and one Input Capture function. Furthermore, it is possible to use the Timer/Counter for Pulse-Width-Modulation (PWM).

The Watchdog Timer is clocked by a separate on-chip oscillator. The Watchdog period can be selected between 16 ms and 2048 ms.

SPI

The Serial Peripheral Interface (SPI) allows synchronous serial high-speed communication.

UART

A comfortable Universal Asynchronous Receiver/Transmitter (UART) allows flexible asynchronous serial communication.

Analog Comparator

The Analog Comparator compares voltages at two pins.

I/O Ports

The AT90S8515 has four I/O ports, which can be operate as digital input or output controlled by the Data Direction Register (DDR). As shown in Figure 5, most pins have alternative functions.

Comparing the pin configuration of the AVR microcontrollers and that of the 8051 microcontroller family reveals one objective of this new microcontroller family.

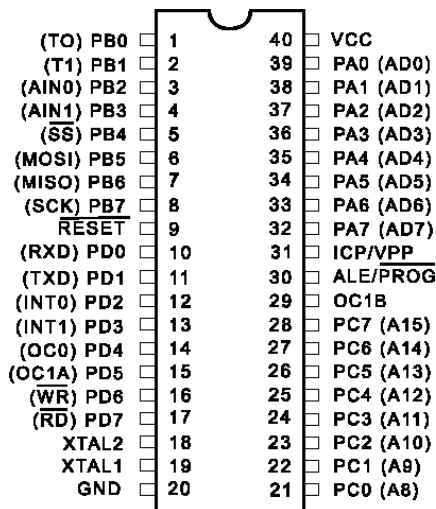


Figure 5 Pin configuration AT90S8515

All I/O ports are bidirectional with individually selectable Pull-up resistors. The outputs can drop to 20 mA so that LEDs can be directly driven.

The AVR microcontrollers support a high-voltage (12 V) parallel programming mode and a low-voltage serial programming mode. The serial programming mode via SPI provides a convenient way to download programs and data into the device inside the user's system.

To get an impression of the instruction set of the AVR microcontrollers, Table 2 explains all instructions in a compact form.

Mnemonics	Description		Cycles
ARITHMETIC AND LOGIC INSTRUCTIONS			
ADD Rd, Rr	Add without Carry	$Rd \leftarrow Rd + Rr$	1
ADC Rd, Rr	Add with Carry	$Rd \leftarrow Rd + Rr + C$	1
ADIW Rd, K	Add Immediate to Word	$Rd+1:Rd \leftarrow Rd+1:Rd + K$	2
SUB Rd, Rr	Subtract without Carry	$Rd \leftarrow Rd - Rr$	1
SUBI Rd, K	Subtract Immediate	$Rd \leftarrow Rd - K$	1
SBC Rd, Rr	Subtract with Carry	$Rd \leftarrow Rd - Rr - C$	1
SBCI Rd, K	Subtract Immediate with Carry	$Rd \leftarrow Rd - K - C$	1
SBIW Rd, K	Subtract Immediate from Word	$Rd+1:Rd \leftarrow Rd+1:Rd - K$	2
AND Rd, Rr	Logical AND	$Rd \leftarrow Rd \bullet Rr$	1
ANDI Rd, K	Logical AND with Immediate	$Rd \leftarrow Rd \bullet K$	1
OR Rd, Rr	Logical OR	$Rd \leftarrow Rd \vee Rr$	1
ORI Rd, K	Logical OR with Immediate	$Rd \leftarrow Rd \vee K$	1
EOR Rd, Rr	Exclusive OR	$Rd \leftarrow Rd \oplus Rr$	1
COM Rd	One's Complement	$Rd \leftarrow \$FF - Rd$	1
NEG Rd	Two's Complement	$Rd \leftarrow \$00 - Rd$	1
SBR Rd,K	Set bit(s) in Register	$Rd \leftarrow Rd \vee K$	1
CBR Rd,K	Clear bit(s) in Register	$Rd \leftarrow Rd \bullet (\$FFh - K)$	1
INC Rd	Increment $Rd \leftarrow Rd + 1$	$Rd \leftarrow Rd + 1$	1
DEC Rd	Decrement	$Rd \leftarrow Rd - 1$	1
TST Rd	Test for Zero or Minus	$Rd \leftarrow Rd \bullet Rd$	1
CLR Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	1
SER Rd	Set Register	$Rd \leftarrow \$FF$	1
MUL Rd,Rr	Multiply Unsigned	$R1, R0 \leftarrow Rd \times Rr$	2
BRANCH INSTRUCTIONS			
RJMP k	Relative Jump	$PC \leftarrow PC + k + 1$	2
IJMP	Indirect Jump to (Z)	$PC \leftarrow Z$	2
JMP k	Jump	$PC \leftarrow k$	3
RCALL k	Relative Call Subroutine	$PC \leftarrow PC + k + 1$	3
ICALL	Indirect Call to (Z)	$PC \leftarrow Z$	3
CALL k	Call Subroutine	$PC \leftarrow k$	4
RET	Subroutine Return	$PC \leftarrow STACK$	4
RETI	Interrupt Return	$PC \leftarrow STACK$	4
CPSE Rd,Rr	Compare, Skip if Equal if (Rd = Rr)	$PC \leftarrow PC + 2$ or 3	1 / 2
CP Rd,Rr	Compare	$Rd - Rr$	1
CPC Rd,Rr	Compare with Carry	$Rd - Rr - C$	1
CPI Rd,K	Compare with Immediate	$Rd - K$	1
SBRC Rr, b	Skip if bit in Register Cleared	if (Rr(b)=0) $PC \leftarrow PC + 2$ or 3	1 / 2
SBRS Rr, b	Skip if bit in Register Set	if (Rr(b)=1) $PC \leftarrow PC + 2$ or 3	1 / 2
SBIC P, b	Skip if bit in I/O Register Cleared	if (I/O(P,b)=0) $PC \leftarrow PC + 2$ or 3	2 / 3
SBIS P, b	Skip if bit in I/O Register Set	If (I/O(P,b)=1) $PC \leftarrow PC + 2$ or 3	2 / 3
BRBS s, k	Branch if Status Flag Set	if (SREG(s) = 1) then $PC \leftarrow PC + k + 1$	1 / 2
BRBC s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then $PC \leftarrow PC + k + 1$	1 / 2

BREQ k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	1 / 2
BRNE k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	1 / 2
BRCS k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	1 / 2
BRCC k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	1 / 2
BRSH k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	1 / 2
BRLO k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	1 / 2
BRMI k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	1 / 2
BRPL k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	1 / 2
BRGE k	Branch if Greater or Equal, Signed	if (N ⊕ V = 0) then PC ← PC + k + 1	1 / 2
BRLT k	Branch if Less Than, Signed	if (N ⊕ V = 1) then PC ← PC + k + 1	1 / 2
BRHS k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	1 / 2
BRHC k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	1 / 2
BRTS k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	1 / 2
BRTC k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	1 / 2
BRVS k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	1 / 2
BRVC k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	1 / 2
BRIE k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	1 / 2
BRID k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	1 / 2

DATA TRANSFER INSTRUCTIONS

MOV Rd, Rr	Copy Register	Rd ← Rr	1
LDI Rd, K	Load Immediate	Rd ← K	1
LDS Rd, k	Load Direct from SRAM	Rd ← (k)	3
LD Rd, X	Load Indirect	Rd ← (X)	2
LD Rd, X+	Load Indirect and Post- Increment	Rd ← (X), X ← X + 1	2
LD Rd, -X	Load Indirect and Pre- Decrement	X ← X - 1, Rd ← (X)	2
LD Rd, Y	Load Indirect	Rd ← (Y)	2
LD Rd, Y+	Load Indirect and Post- Increment	Rd ← (Y), Y ← Y + 1	2
LD Rd, -Y	Load Indirect and Pre- Decrement	Y ← Y - 1, Rd ← (Y)	2
LDD Rd, Y+q	Load Indirect with Displacement	Rd ← (Y + q)	2
LD Rd, Z	Load Indirect	Rd ← (Z)	2

LD Rd, Z+	Load Indirect and Post-Increment	$Rd \leftarrow (Z), Z \leftarrow Z+1$	2
LD Rd, -Z	Load Indirect and Pre-Decrement	$Z \leftarrow Z - 1, Rd \leftarrow (Z)$	2
LDD Rd, Z+q	Load Indirect with Displacement	$Rd \leftarrow (Z + q)$	2
STS k, Rr	Store Direct to SRAM	$Rd \leftarrow (k)$	3
ST X, Rr	Store Indirect	$(X) \leftarrow Rr$	2
ST X+, Rr	Store Indirect and Post-Increment	$(X) \leftarrow Rr, X \leftarrow X + 1$	2
ST -X, Rr	Store Indirect and Pre-Decrement	$X \leftarrow X - 1, (X) \leftarrow Rr$	2
ST Y, Rr	Store Indirect	$(Y) \leftarrow Rr$	2
ST Y+, Rr	Store Indirect and Post-Increment	$(Y) \leftarrow Rr, Y \leftarrow Y + 1$	2
ST -Y, Rr	Store Indirect and Pre-Decrement	$Y \leftarrow Y - 1, (Y) \leftarrow Rr$	2
STD Y+q,Rr	Store Indirect with Displacement	$(Y + q) \leftarrow Rr$	2
ST Z, Rr	Store Indirect	$(Z) \leftarrow Rr$	2
ST Z+, Rr	Store Indirect and Post-Increment	$(Z) \leftarrow Rr, Z \leftarrow Z + 1$	2
ST -Z, Rr	Store Indirect and Pre-Decrement	$Z \leftarrow Z - 1, (Z) \leftarrow Rr$	2
STD Z+q,Rr	Store Indirect with Displacement	$(Z + q) \leftarrow Rr$	2
LPM	Load Program Memory	$R0 \leftarrow (Z)$	3
IN Rd, P	In Port	$Rd \leftarrow P$	1
OUT P, Rr	Out Port	$P \leftarrow Rr$	1
PUSH Rr	Push Register on Stack	$STACK \leftarrow Rr$	2
POP Rd	Pop Register from Stack	$Rd \leftarrow STACK$	2
BIT AND BIT-TEST INSTRUCTIONS			
LSL Rd	Logical Shift Left	$Rd(n+1) \leftarrow Rd(n),$ $Rd(0) \leftarrow 0, C \leftarrow Rd(7)$	1
LSR Rd	Logical Shift Right	$Rd(n) \leftarrow Rd(n+1),$ $Rd(7) \leftarrow 0, C \leftarrow Rd(0)$	1
ROL Rd	Rotate Left Through Carry	$Rd(0) \leftarrow C,$ $Rd(n+1) \leftarrow Rd(n), C \leftarrow Rd(7)$	1
ROR Rd	Rotate Right Through Carry	$Rd(7) \leftarrow C,$ $Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)$	1
ASR Rd	Arithmetic Shift Right	$Rd(n) \leftarrow Rd(n+1), n=0..6$	1
SWAP Rd	Swap Nibbles	$Rd(3..0) \leftrightarrow Rd(7..4)$	1
BSET s	Flag Set	$SREG(s) \leftarrow 1$	1
BCLR s	Flag Clear	$SREG(s) \leftarrow 0$	1
SBI P, b	Set bit in I/O Register	$I/O(P, b) \leftarrow 1$	2
CBI P, b	Clear bit in I/O Register	$I/O(P, b) \leftarrow 0$	2
BST Rr, b	bit Store from Register to T	$T \leftarrow Rr(b)$	1
BLD Rd, b	bit load from T to Register	$Rd(b) \leftarrow T$	1
SEC	Set Carry	$C \leftarrow 1$	1
CLC	Clear Carry	$C \leftarrow 0$	1
SEN	Set Negative Flag	$N \leftarrow 1$	1
CLN	Clear Negative Flag	$N \leftarrow 0$	1

SEZ	Set Zero Flag	$Z \leftarrow 1$	1
CLZ	Clear Zero Flag	$Z \leftarrow 0$	1
SEI	Global Interrupt Enable	$I \leftarrow 1$	1
CLI	Global Interrupt Disable	$I \leftarrow 0$	1
SES	Set Signed Test Flag	$S \leftarrow 1$	1
CLS	Clear Signed Test Flag	$S \leftarrow 0$	1
SEV	Set Two's Complement Overflow	$V \leftarrow 1$	1
CLV	Clear Two's Complement Overflow	$V \leftarrow 0$	1
SET	Set T in SREG	$T \leftarrow 1$	1
CLT	Clear T in SREG	$T \leftarrow 0$	1
SEH	Set Half Carry Flag in SREG	$H \leftarrow 1$	1
CLH	Clear Half Carry Flag in SREG	$H \leftarrow 0$	1
NOP	No Operation	None	1
SLEEP	Sleep		1
WDR	Watchdog Reset		1

Table 2 Instruction Set of AVR microcontrollers

These introducing remarks on the AVR microcontrollers cannot of course replace a detailed study of the technical documentation of the manufacturer. Descriptions of the individual microcontrollers as well as application notes and program examples can be found on Atmel's web site [<http://www.atmel.com>]. The manufacturer's documentation is complemented by further publications [3][4].

2 BASCOM

BASCOM-AVR is not only a BASIC Compiler, but also a comfortable Integrated Development Environment (IDE) running under Windows 95 and Windows NT.

Such a development environment supports the whole process from coding and testing a program to programming the used micro-controller.

In this book the term BASCOM is used when no distinction must be made between BASCOM-8051 and BASCOM-AVR. In all cases where a distinction is necessary, a few changes only are required to make the program work with the other family of microcontrollers. This is one important advantage of high-level languages.

So as to prevent that work with BASCOM and the program examples in this book are mere dry homework, a demo of BASCOM-8051 or BASCOM-AVR can be used for first tests. These BASCOM demos can be downloaded free of charge from different URLs.

For proper installation of the required BASCOM IDE, make sure a printer is installed - the printer need not necessarily be used or connected.

The licence agreement must be accepted before one of the BASCOM IDEs is installed

2.1 *BASCOM Demos*

Over a link to the download area of the BASCOM developer MCS Electronics [<http://www.mcselec.com>] some files are available for download.

For download the BASCOM-8051 demo use this URL

http://www.mcselec.com/download_8051.htm

and for downloading the BASCOM-AVR demo use

http://www.mcselec.com/download_avr.htm .

On these download sites you will find the manuals as PDF and all information required for an upgrade to the commercial versions.

After extracting all downloaded files to a separate directory, there is a setup program for installation.

Installation starts as usual under Windows when this setup program is called.

After completion of the installation, the following files need to be installed on the PC. Figure 6 shows the files installed for BASCOM-AVR as an example. Inspecting the directory with the Explorer will show some more files there. These files will be explained later.



Figure 6 BASCOM-AVR Demo Files

As is common for most demo programs, some restrictions must be expected. The only restriction of both BASCOM demos is a reduced code size of 2 KB.

If the code size exceeds this limit after compilation, the compiler will generate error messages as shown in Figure 7.

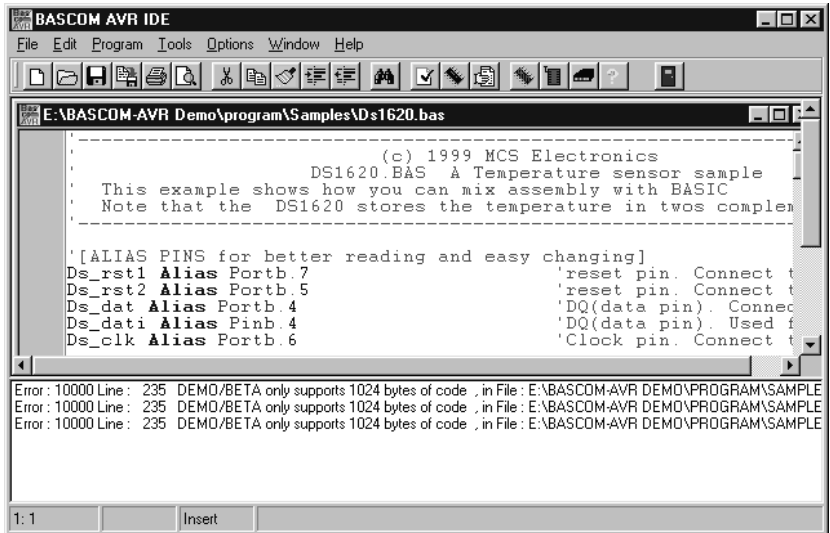


Figure 7 Error messages due to exceeding the restricted code size

2.2 BASCOM Commercial Versions

If you decide to buy the commercial version of the used BASCOM IDE, you may order it from <http://www.mcselec.com> or one of the local distributors. Downloading the files and ordering the license is done in next to no time. The license will be sent immediately by e-mail.

The installation of the commercial version does not differ from the procedure for the BASCOM demo. Start Setup and follow the instructions of the Setup program.

2.3 Update of BASCOM Commercial Versions

When a commercial version of BASCOM is installed, it can be updated when a new version is ready for downloading from MCS Electronic's web site. In the download area you will find a link to an AutoUpdate program.

Install this AutoUpdate program in your BASCOM-8051 or BASCOM-AVR subdirectory as you installed BASCOM-8051 or BASCOM-AVR before.