


Micro-Code


Der gesamte Micro-Code, d.h. der Inhalt der Micro-Code-ROMs, ist in der Datei 65C02µC.PDF aufgelistet.

Der Micro-Code jedes einzelnen Befehls kann man sich in der Projektdatei 65C02µC.prj schrittweise gezielt anzeigen lassen. Auf den hier folgenden Seiten werden für jeden Befehl bzw. Befehlsabschnitt alle Teilschritte erläutert.

In der Kopfzeile steht der Befehlsname und rechts das hexadezimale Befehlsbyte bzw. bei Sequenzabschnitten die Funktionsbezeichnung. Bei Befehlen folgt der Überblick über die durch den Befehl beeinflussten CCR-Bits und ggfs. eine Assembleranweisung als Beispiel.

Darunter ist der aktuelle Stand des Programmzählers (relativ zum Befehlsbyte), der Stand des Einzelschrittzählers (µStep-Counter) und als wesentliche Information die Steuersignale jedes Einzelschritts mit Kommentar aufgeführt.

Alle Signale, die bei dem jeweiligen µStep-Zählerstand mit steigender Flanke aktiv werden, sind durch die grafische Ergänzung  gekennzeichnet.

Alle Signale, die bei dem jeweiligen µStep-Zählerstand mit fallender Flanke aktiv werden, sind durch die grafische Ergänzung  gekennzeichnet.

Bis auf wenige Ausnahmen (Sequenzsprung bei bedingten Sprüngen) endet jede Teilschrittsequenz mit den Fetch-Cycles F1 und F2, die den nächsten Befehl in das Pre-Command-Register laden und dekodieren. Da der Adressgenerator und die ALU/SHIFT-Unit durch einen Tri-State-Baustein vom externen Datenbus abgekoppelt werden können, ist es bei Befehlen, die keine Ergebnisse über den Datenbus zum Arbeitsspeicher senden, möglich, den Fetch-Cycle F1 schon während der Ausführungsphase abzuarbeiten, wodurch bei diesen Befehlen ein Takt eingespart werden kann (Fetch while Execution).

Der Micro-Code ist für alle Sonderbefehle (P1-Befehle), Ein-Byte-Befehle, absolute Sprungbefehle und bedingte bzw. nicht bedingte Sprungbefehle explizit aufgeführt.

FETCH (F1 & F2)

FETCH

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
next cmd.	xxxx	[FETCH° PC TO AB°]	indicate Fetch-Cycle (read next command) PC to AB
		[-]	(command is clocked into command- register automatically)
next cmd	xxxx	[FETCH°]	indicate Fetch-Cycle (decode command)
		[CLK PC°]	clock PC to next command or argument
next cmd.			

RESET

1000

HEX	ASSEMBLER INSTR.	CCR
—	— —	- - - 0 1 - 0
—		D I C

PC	μCNT	μSTEP	COMMENT
undef.	1000	[-	nothing to synchronize with clock
		[-	nothing to synchronize with clock
undef.	1001	[MUX CCR to 001 VECTORS to 011 SP LD°	"1" to CCR-Bits SP-value "FF" to LB prepare SP-loading
		[CLK I° CLK SP°	disable IRQ set SP
undef.	1002	[MUX CCR to 000 VECTORS to 101 AR LD°	"0" to CCR-Bits "FFFC" to bus prepare AR-loading
		[CLK C° CLK D° CLK AR°	clear C-Bit disable Decimal-Mode set AR
undef.	1003	[AR TO AB° ENL°	AR to AB to read PC LB via low bus
		[CLK TEMP°	into TEMP
undef.	1004	[-	
		[CLK AR°	increment AR to PC HB
undef.	1005	[AR TO AB° ENH° TEMP TO BUS° PC LD°	AR to AB to read PC HB via high bus PC LB to low bus prepare PC-loading
		[CLK PC°	into PC
new	1006	{FETCH READ COMMAND}	
PC	1007	{FETCH DECODE COMMAND}	

BRK

Command 00 / Execution at 1010

HEX	ASSEMBLER INSTR.	CCR
00	BRK	- - - 1 - 1 - -
—		B I

PC	μCNT	μSTEP	COMMENT
0001	1010	[
		CLK PC°	for compatibility set PC to wrong return-address
0002	1011	[
		PC TO BUS°	to save actual
		ENH°	PC HB
		I/O°	out to
		SP TO AB°	stack
		SP U/D°	prepare decrementation of stackpointer
		MUX CCR to 001	"1" to CCR-Bits
		[
		R/W°	write PC HB to stack
		CLK I°	disable interrupt
		CLK B°	indicate BRK-command
		CLK SP°	decrement stackpointer to next address
0002	1012	[
		PC TO BUS°	to save actual
		ENL°	PC LB
		I/O°	out to
		SP TO AB°	stack
		SP U/D°	prepare decrementation of stackpointer
		[
		R/W°	write PC HB to stack
		CLK SP°	decrement stackpointer to next address
0002	1013	[
		CCR TO BUS°	to save CCR
		ENL°	via low bus
		I/O°	out to
		SP TO AB°	stack
		SP U/D°	prepare decrementation of stackpointer
		[
		R/W°	write PC HB to stack
		CLK SP°	decrement stackpointer to next address
0002	1014	[
		VECTORS to 110	INT/BRK-Vector "FFFE"
		AR LD°	into
		[
		CLK AR°	AR
0002	1015	[
		AR TO AB°	read INT/BRK-Vector LB
		ENL°	via low bus
		[
		CLK TEMP°	into TEMP
		CLK AR°	increment AR to "FFFF"

BRK**continued**

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0002	1016	[TEMP TO BUS° AR TO AB° ENH° PC LD°]	INT/BRK-Vector LB to low bus read INT/BRK-Vector HB via high bus prepare PC-loading
		[CLK PC°]	load new PC
BRK-	1017	{FETCH READ COMMAND}	
Address	1018	{FETCH DECODE COMMAND}	

NMI

1020

PC	PCNT	STEP	COMMENT
old PC	1020	<div> <div>PC TO BUS°</div> <div>ENH°</div> <div>I/O°</div> <div>SP TO AB°</div> <div>SP U/D°</div> </div>	<div> <div>save PC</div> <div>HB</div> <div>out</div> <div>to SP</div> <div>prepare decrementing SP</div> </div>
		<div> <div>R/W°</div> <div>CLK SP°</div> </div>	<div> <div>write PC HB to SP</div> <div>set SP to next address</div> </div>
old PC	1021	<div> <div>PC TO BUS°</div> <div>ENL°</div> <div>I/O°</div> <div>SP TO AB°</div> <div>SP U/D°</div> </div>	<div> <div>save PC</div> <div>LB</div> <div>out</div> <div>to SP to</div> <div>prepare decrementing SP</div> </div>
		<div> <div>R/W°</div> <div>CLK SP°</div> </div>	<div> <div>write PC LB to SP</div> <div>set SP to next address</div> </div>
old PC	1022	<div> <div>CCR TO BUS°</div> <div>ENL°</div> <div>I/O°</div> <div>SP TO AB°</div> <div>SP U/D°</div> </div>	<div> <div>save CCR</div> <div>LB</div> <div>out</div> <div>to SP to</div> <div>prepare decrementing SP</div> </div>
		<div> <div>R/W°</div> <div>CLK SP°</div> </div>	<div> <div>write PC LB to SP</div> <div>set SP to next address</div> </div>
old PC	1023	<div> <div>AR LD°</div> <div>VECTORS to 100</div> </div>	<div> <div>load</div> <div>NMI-Vector to "FFFA"</div> </div>
		<div> <div>CLK AR°</div> </div>	<div> <div>into AR</div> </div>
old PC	1024	<div> <div>AR TO AB°</div> <div>ENL</div> </div>	<div> <div>read new PC</div> <div>LB</div> </div>
		<div> <div>CLK TEMP°</div> <div>CLK AR°</div> </div>	<div> <div>into TEMP</div> <div>set AR NMI-Vector to "FFFFB"</div> </div>
old PC	1025	<div> <div>AR TO AB°</div> <div>ENH°</div> <div>TEMP TO BUS°</div> <div>PC LD°</div> </div>	<div> <div>read new PC</div> <div>HB</div> <div>get saved LB</div> <div>prepare loading PC</div> </div>
		<div> <div>CLK PC°</div> </div>	<div> <div>load PC</div> </div>
NMI- Address	1026 1027	<div> <div>{FETCH READ COMMAND}</div> <div>{FETCH DECODE COMMAND}</div> </div>	

INT

1040

PC	μCNT	μSTEP	COMMENT
0001	1040	<div> <div>PC TO BUS°</div> <div>ENH°</div> <div>I/O°</div> <div>SP TO AB°</div> <div>SP U/D°</div> <div>MUX CCR to 001</div> </div>	<div>to save actual</div> <div>PC HB</div> <div>out to</div> <div>stack</div> <div>prepare decrementation of stackpointer</div> <div>"1" to CCR-Bits</div>
		<div> <div>R/W°</div> <div>CLK I°</div> <div>CLK SP°</div> </div>	<div>write PC HB to stack</div> <div>disable interrupt</div> <div>decrement stackpointer to next address</div>
0001	1041	<div> <div>PC TO BUS°</div> <div>ENL°</div> <div>I/O°</div> <div>SP TO AB°</div> <div>SP U/D°</div> </div>	<div>to save actual</div> <div>PC LB</div> <div>out to</div> <div>stack</div> <div>prepare decrementation of stackpointer</div>
		<div> <div>R/W°</div> <div>CLK SP°</div> </div>	<div>write PC HB to stack</div> <div>decrement stackpointer to next address</div>
0001	1042	<div> <div>CCR TO BUS°</div> <div>ENL°</div> <div>I/O°</div> <div>SP TO AB°</div> <div>SP U/D°</div> </div>	<div>to save CCR</div> <div>via low bus</div> <div>out to</div> <div>stack</div> <div>prepare decrementation of stackpointer</div>
		<div> <div>R/W°</div> <div>CLK SP°</div> </div>	<div>write PC HB to stack</div> <div>decrement stackpointer to next address</div>
0001	1043	<div> <div>VECTORS to 110</div> <div>AR LD°</div> </div>	<div>INT-Vector "FFFE"</div> <div>into</div>
		<div> <div>CLK AR°</div> </div>	<div>AR</div>
0001	1044	<div> <div>AR TO AB°</div> <div>ENL°</div> </div>	<div>read INT-Vector LB</div> <div>via low bus</div>
		<div> <div>CLK TEMP°</div> <div>CLK AR°</div> </div>	<div>into TEMP</div> <div>increment AR to "FFFF"</div>
0001	1045	<div> <div>TEMP TO BUS°</div> <div>AR TO AB°</div> <div>ENH°</div> <div>PC LD°</div> </div>	<div>INT-Vector LB to low bus</div> <div>read INT-Vector HB</div> <div>via high bus</div> <div>prepare PC-loading</div>
		<div> <div>CLK PC°</div> </div>	<div>load new PC</div>
INT-Address	1046	{FETCH READ COMMAND}	
	1047	{FETCH DECODE COMMAND}	

HEX	ASSEMBLER INSTR.	CCR
02	SWI 0A	- - - - 1 - - -
—		I

PC	μCNT	μSTEP	COMMENT
0001 next cmd.	0020	<div>PC to AB°</div> <div>ENL°</div> <div>MUX CCR to 011</div>	<div>get SWI-No. (= SWI-Vector-LB)</div> <div>prepare disabling interrupt</div>
		<div>CLK TEMP°</div> <div>CLK PC°</div> <div>CLK I-BIT°</div>	<div>save SWI-Vector-Address LB to TEMP</div> <div>increment PC to next command</div> <div>disable interrupt</div>
	0021	<div>PC to BUS°</div> <div>SP to AB°</div> <div>ENH°</div> <div>I/O°</div> <div>SP U/D°</div>	save PC(HB) to stack
		<div>CLK SP°</div> <div>R/W°</div>	
	0022	<div>PC to BUS°</div> <div>SP to AB°</div> <div>ENL°</div> <div>I/O°</div> <div>SP U/D°</div>	save PC(LB) to stack
		<div>CLK SP°</div> <div>R/W°</div>	
	0023	<div>CCR to BUS°</div> <div>SP to AB°</div> <div>ENL°</div> <div>I/O°</div> <div>SP U/D°</div>	save CCR to stack
		<div>CLK SP°</div> <div>R/W°</div>	
	0024	<div>VECTOR to 010</div> <div>TEMP to BUS°</div> <div>AR LD°</div>	<div>set SWI-Vector-Address HB to Page-2</div> <div>activate SWI-Vector-Address LB</div> <div>load complete SWI-Vector-Address</div>
		<div>CLK AR°</div>	into AR

SWI xx

continued

PC	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
	0025	<div> <div>AR to AB°</div> <div>ENL°</div> </div>	get SWI-Vector LB
		<div> <div>CLK TEMP°</div> <div>CLK AR°</div> </div>	<div>into TEMP</div> <div>increment AR to SWI-Vector HB</div>
	0026	<div> <div>AR to AB°</div> <div>ENH°</div> <div>TEMP to BUS°</div> <div>PCLD°</div> </div>	<div>get SWI-Vector-HB</div> <div>activate SWI-Vector LB</div> <div>prepare loading of PC</div>
		<div> <div>CLK PC°</div> </div>	load complete SWI-Vector into PC
SWI -	0027	{FETCH READ COMMAND}	
Vector	0028	{FETCH DECODE COMMAND}	

BRA unconditional rel. 8-Bit-Offset

80

HEX	ASSEMBLER INSTR.	CCR
80	BRA #07	- - - - -
07		

PC	μCNT	μSTEP	COMMENT
0001	0800		
offset		PC TO AB°	PC to AB
		ENL°	to read offset (07h)
		CLK TEMP°	into TEMP
		CLK PC°	increment PC to next command
0002	0801		
		TEMP TO BUS°	offset to Adder LB A
		VECTOR to 001	00h/FFh (R8) to Adder HB A
		PC to ADD°	PC to Adder B (LB, HB)
		PC FROM BUS/ADD°	result of ADD to PC
		PC LD°	prepare PC for loading
		CLK PC°	load PC with new calculated address
0002	0802	{FETCH READ COMMAND}	
	0803	{FETCH DECODE COMMAND}	

BRN conditional rel. 8-Bit-Offset

XX

μSteps valid for all following commands :

HEX	ASSEMBLER INSTR.	CCR
90	BCC	- - - - -
B0	BCS	- - - - -
F0	BEQ	- - - - -
30	BMI	- - - - -
D0	BNE	- - - - -
10	BPL	- - - - -
50	BVC	- - - - -
70	BVS	- - - - -

HEX	ASSEMBLER INSTR.	CCR
10	BPL #08	- - - - -
08		

do following μSteps if branch is not performed

PC	μCNT	μSTEP	COMMENT
0001 (Off-set LB)	0XX0	[nothing to do if no branch is performed
		CLK PC°	increment PC to next command
0XX1		{FETCH READ COMMAND}	
0XX2		{FETCH DECODE COMMAND}	

do following μSteps if branch is performed

PC	μCNT	μSTEP	COMMENT
0001 (Off-set LB)	1080	[PC TO AB°
		ENL°	PC to AB to read offset LB
		CLK TEMP°	into TEMP
		CLK PC°	increment PC to next command for correct addition
0002 (next cmd)	1081	[PC TO ADD°
		TEMP TO BUS°	PC to Adder-B LB/HB
		VECTORS TO 001	TEMP to Adder-A LB
		PC FROM BUS/ADD°	00/FF to Adder-A HB
		PC LD°	Adder result to PC
			prepare PC-loading
		CLK PC°	load new PC
1082		{FETCH READ COMMAND}	
1083		{FETCH DECODE COMMAND}	

HEX	ASSEMBLER INSTR.	CCR
83	83 #0F FF	- - - - -
0F		
FF		

PC	PCNT	μSTEP	COMMENT
0001	0830	[PC TO AB° ENL°	PC to AB to read offset (LB) via low bus
offset LB			
		[CLK TEMP° CLK PC°	save offset (LB) increment PC to offset (HB)
0002	0831	[PC TO AB° ENH° TEMP TO BUS° PC TO ADD° PC FROM ADD° PC LD°	PC to AB to read offset (HB) via high bus to Adder Channel A (HB) offset (LB) to Adder Channel A (LB) PC to Adder Channel B (LB and HB) result to PC prepare PC-loading
offset HB			
		[CLK PC°	load new address into PC
0002	0832	[-	do nothing
		[CLK PC°	increment PC to next command
	0833	{FETCH READ COMMAND}	
	0834	{FETCH DECODE COMMAND}	

BRN conditional rel. 16-bit-Offset

XX

μSteps valid for all following commands :

HEX	ASSEMBLER INSTR.	CCR
93	BCC	- - - - -
B3	BCS	- - - - -
F3	BEQ	- - - - -
33	BMI	- - - - -
D3	BNE	- - - - -
13	BPL	- - - - -
53	BVC	- - - - -
73	BVS	- - - - -

HEX	ASSEMBLER INSTR.	CCR
13	BPL #0108	- - - - -
08		
01		

do following μSteps if branch is not performed

PC	μCNT	μSTEP	COMMENT
0001 (Off- set LB)	0XX0	[nothing to do if no branch is performed
		CLK PC°	increment PC to next command
	0XX1	{FETCH READ COMMAND}	
	0XX2	{FETCH DECODE COMMAND}	

do following μSteps if branch is performed

PC	μCNT	μSTEP	COMMENT
0001 (Off- set LB)	1100	[PC TO AB° ENL°
			PC to AB to read offset LB
		CLK TEMP° CLK PC°	into TEMP increment PC to Offset HB
0002 Offset HB	1101	[PC TO AB° ENH° TEMP TO BUS° PC TO ADD° PC FROM BUS/ADD° PC LD°
			PC to AB to read offset HB TEMP to Adder-A LB PC to Adder-B LB/HB Adder result to PC prepare PC-loading
		CLK PC°	load new PC-1
new PC-1	1102	[
		CLK PC°	adjust PC (increment wrong PC)
	1103	{FETCH READ COMMAND}	
	1104	{FETCH DECODE COMMAND}	

BBR ZP, rel. 8-Bit-Offset

XX

HEX	ASSEMBLER INSTR.	CCR
0F	BBR0 \$\$, OFFSET8	- - - - -
1F	BBR1 \$\$, OFFSET8	- - - - -
2F	BBR2 \$\$, OFFSET8	- - - - -
3F	BBR3 \$\$, OFFSET8	- - - - -
4F	BBR4 \$\$, OFFSET8	- - - - -
5F	BBR5 \$\$, OFFSET8	- - - - -
6F	BBR6 \$\$, OFFSET8	- - - - -
7F	BBR7 \$\$, OFFSET8	- - - - -

PC	μCNT	μSTEP	COMMENT
0001	0xx0	[PC TO AB° ENL° VECTORS to 000 AR LD°	PC to AB to read ZP-Address via low bus 00h to bus HB prepare AR-loading
		[CLK AR° CLK PC°	ZP-Address into AR increment PC to Offset
0001	0xx1	[AR TO AB° ENL°	AR to AB to read byte to operate on via low bus
		[CLK ALU-IN°	into ALU-IN
0002	0xx2	[ALU-IN TO ALU-B° RMB/BBR° BIT-BRN-CMD° ALU to 10001	ALU-IN to ALU-B appropriate bit-mask to ALU-A indicate Branch-On-Bit-Command set ALU to A NOR B
0002	0xx3	[execute when no branch
		[CLK PC°	increment PC to next command
	0xx4	{FETCH READ COMMAND}	execute when no branch
	0xx5	{FETCH DECODE COMMAND}	execute when no branch

If branch is performed, μSteps 1080 - 1083 will be executed (see BRN8)

BBR ZP, rel. 16-Bit-Offset

XX

HEX	ASSEMBLER INSTR.	CCR
0B	BBR0 \$\$, OFFSET16	- - - - -
1B	BBR1 \$\$, OFFSET16	- - - - -
2B	BBR2 \$\$, OFFSET16	- - - - -
3B	BBR3 \$\$, OFFSET16	- - - - -
4B	BBR4 \$\$, OFFSET16	- - - - -
5B	BBR5 \$\$, OFFSET16	- - - - -
6B	BBR6 \$\$, OFFSET16	- - - - -
7B	BBR7 \$\$, OFFSET16	- - - - -

PC	μCNT	μSTEP	COMMENT
0001	0xx0	PC TO AB° ENL° VECTORS to 000 AR LD°	PC to AB to read ZP-Address via low bus 00h to bus HB prepare AR-loading
		CLK AR° CLK PC°	ZP-Address into AR increment PC to Offset
0001	0xx1	AR TO AB° ENL°	AR to AB to read byte to operate on via low bus
		CLK ALU-IN°	into ALU-IN
0002	0xx2	ALU-IN TO ALU-B° RMB/BBR° BIT-BRN-CMD° ALU to 10001	ALU-IN to ALU-B appropriate bit-mask to ALU-A indicate Branch-On-Bit-Command set ALU to A NOR B
0002	0xx3		execute when no branch
Offset LB		CLK PC°	increment PC to Offset HB
0003	0xx4		execute when no branch
Offset HB		CLK PC°	increment PC to next command
	0xx5	{FETCH READ COMMAND}	execute when no branch
	0xx6	{FETCH DECODE COMMAND}	execute when no branch

If branch is performed, μSteps 1100 - 1103 will be executed (see BRN16)

BBS ZP, rel. 8-Bit-Offset

XX

HEX	ASSEMBLER INSTR.	CCR
8F	BBS0 \$\$, OFFSET8	- - - - -
9F	BBS1 \$\$, OFFSET8	- - - - -
AF	BBS2 \$\$, OFFSET8	- - - - -
BF	BBS3 \$\$, OFFSET8	- - - - -
CF	BBS4 \$\$, OFFSET8	- - - - -
DF	BBS5 \$\$, OFFSET8	- - - - -
EF	BBS6 \$\$, OFFSET8	- - - - -
FF	BBS7 \$\$, OFFSET8	- - - - -

PC	μCNT	μSTEP	COMMENT
0001	0xx0	[PC TO AB° ENL° VECTORS to 000 AR LD°	PC to AB to read ZP-Address via low bus 00h to bus HB prepare AR-loading
		[CLK AR° CLK PC°	ZP-Address into AR increment PC to Offset
0001	0xx1	[AR TO AB° ENL°	AR to AB to read byte to operate on via low bus
		[CLK ALU-IN°	into ALU-IN
0002	0xx2	[ALU-IN TO ALU-B° SMB/BBS° BIT-BRN-CMD° ALU to 11011	ALU-IN to ALU-B appropriate bit-mask to ALU-A indicate Branch-On-Bit-Command set ALU to A AND B
0002	0xx3	[execute when no branch
		[CLK PC°	increment PC to next command
	0xx4	{FETCH READ COMMAND}	execute when no branch
	0xx5	{FETCH DECODE COMMAND}	execute when no branch

If branch is performed, μSteps 1080 - 1083 will be executed (see BRN8)

BBS ZP, rel. 16-Bit-Offset

XX

HEX	ASSEMBLER INSTR.	CCR
8B	BBS0 \$\$, OFFSET16	- - - - -
9B	BBS1 \$\$, OFFSET16	- - - - -
AB	BBS2 \$\$, OFFSET16	- - - - -
BB	BBS3 \$\$, OFFSET16	- - - - -
CB	BBS4 \$\$, OFFSET16	- - - - -
DB	BBS5 \$\$, OFFSET16	- - - - -
EB	BBS6 \$\$, OFFSET16	- - - - -
FB	BBS7 \$\$, OFFSET16	- - - - -

PC	μCNT	μSTEP	COMMENT
0001	0xx0	[PC TO AB° ENL° VECTORS to 000 AR LD°	PC to AB to read ZP-Address via low bus 00h to bus HB prepare AR-loading
		[CLK AR° CLK PC°	ZP-Address into AR increment PC to Offset
0001	0xx1	[AR TO AB° ENL°	AR to AB to read byte to operate on via low bus
		[CLK ALU-IN°	into ALU-IN
0002	0xx2	[ALU-IN TO ALU-B° SMB/BBS° BIT-BRN-CMD° ALU to 11011	ALU-IN to ALU-B appropriate bit-mask to ALU-A indicate Branch-On-Bit-Command set ALU to A AND B
0002	0xx3	[execute when no branch
Offset LB		[CLK PC°	increment PC to Offset HB
0003	0xx4	[execute when no branch
Offset HB		[CLK PC°	increment PC to next command
	0xx5	{FETCH READ COMMAND}	execute when no branch
	0xx6	{FETCH DECODE COMMAND}	execute when no branch

If branch is performed, μSteps 1100 - 1103 will be executed (see BRN16)

JMP abs.

4C

HEX	ASSEMBLER INSTR.	CCR
4C	JMP \$0208	- - - - -
08		
02		

PC	PCNT	μSTEP	COMMENT
0001	04C0	[PC TO AB° ENL°	PC to AB to read abs. address LB via low bus (08h)
LB abs. Address			
		[CLK TEMP° CLK PC°	into TEMP increment PC to abs. address HB
0002	04C1	[PC TO AB° ENH° TEMP TO BUS° PC LD°	PC to AB to read abs. Address HB via high bus (02h) abs. address LB to low bus (08h) prepare PC-loading
HB abs. Address			
		[CLK PC°	write Abs. Address into PC
	04C2	{FETCH READ COMMAND}	
	04C3	{FETCH DECODE COMMAND}	

JMP (abs.)

6C

HEX	ASSEMBLER INSTR.	CCR
6C	JMP (\$0208)	- - - - -
08		
02		

PC	μCNT	μSTEP	COMMENT
0001	06C0	[PC TO AB° ENL°	PC to AB to read ind. address LB via low bus
LB ind. Address		[CLK TEMP° CLK PC°	into TEMP increment PC to ind. address HB
0002	06C1	[PC TO AB° ENH° TEMP TO BUS° AR LD°	PC to AB to read ind. Address HB via high bus ind. Address LB to low bus prepare AR-loading
HB ind. Address		[CLK AR°	write ind. Address into AR
0002	06C2	[AR TO AB° ENL°	AR to AB to read abs. address LB via low bus
HB ind. Address		[CLK TEMP° CLK AR°	into TEMP increment AR to abs. address HB
0002	06C3	[AR TO AB° ENH° TEMP TO BUS° PC LD°	AR to AB to read abs. Address HB via high bus abs. Address LB to low bus prepare PC-loading
HB ind. Address		[CLK PC°	write Abs. Address into PC
06C4	{FETCH READ COMMAND}		
06C5	{FETCH DECODE COMMAND}		

JMP (abs.,X)

7C

HEX	ASSEMBLER INSTR.	CCR
7C	JMP (\$0208,X)	- - - - -
08		
02		

PC	μCNT	μSTEP	COMMENT
0001	07C0	[
LB ind.		PC TO AB°	PC to AB
Address		ENL°	to read ind. address LB via low bus
]	
		CLK TEMP°	into TEMP
		CLK PC°	increment PC to ind. address HB
0002	07C1	[
HB ind.		PC TO AB°	PC to AB
Address		ENH°	to read ind. Address HB to Adder HB A
		TEMP TO BUS°	ind. Address LB to Adder LB A
		X TO ADD°	X-REG TO ADDER LB B
		"00" TO ADDER HB B	Adder HB B to 00
		AR FROM BUS/ADD°	Adder result to AR
		AR LD°	prepare AR-loading
]	
		CLK AR°	write ind. Address + X into AR
0002	07C2	[
HB ind.		AR TO AB°	AR to AB
Address		ENL°	to read abs. address LB via low bus
]	
		CLK TEMP°	into TEMP
		CLK AR°	increment AR to abs. address HB
0002	07C3	[
HB ind.		AR TO AB°	AR to AB
Address		ENH°	to read abs. Address HB via high bus
		TEMP TO BUS°	abs. Address LB to low bus
		PC LD°	prepare PC-loading
]	
		CLK PC°	write Abs. Address into PC
07C4	{	FETCH READ COMMAND}	
07C5	{	FETCH DECODE COMMAND}	

JMP (ZP)

DC

HEX	ASSEMBLER INSTR.	CCR
DC	JMP (\$40)	- - - - -
40		

PC	μCNT	μSTEP	COMMENT
0001	0DC0	PC TO AB° ENL° VECTOR to 000 AR LD°	PC to AB to read ZP-Address via low bus 00h to bus HB prepare AR-loading
		CLK AR°	ZP-Address to read into AR
0001	0DC1	AR TO AB° ENL°	AR to AB to read low byte of indirect address
		CLK TEMP°	into TEMP
0001	0DC2	-	
		CLK AR°	increment AR to next ZP-Address (HB)
0002	0DC3	TEMP TO BUS° AR TO AB° ENH° PC LD°	Ind. Address LB to low bus ZP-Address to AB Ind. Address HB via high bus prepare PC-loading of complete indirect address
next cmd		CLK PC°	into PC
	0DC4	{FETCH READ COMMAND}	
	0DC5	{FETCH DECODE COMMAND}	

JMP (ZP,X)

FC

HEX	ASSEMBLER INSTR.	CCR
FC	JMP (\$40,X)	- - - - -
40		

PC	μCNT	μSTEP	COMMENT
0001	0FC0	PC TO AB° ENL°	PC to AB to read ZP-Address LB via low bus to Adder LB A
ZP-		VECTOR to 000	00h to Adder HB A
Addr.		X TO ADD°	X to Adder LB B
		HB-ADD 00°	00 to Adder HB B
		ADDER C/C°	w/o Adder-Carry
		AR FROM BUS/ADD°	Adder-result to AR
		AR LD°	prepare AR-loading
		CLK AR°	(ZP + X) Address to read from into AR
0001	0FC1	AR TO AB° ENL°	AR to AB to read indirect address LB
		CLK TEMP°	into TEMP
0001	0FC2	-	
		CLK AR°	increment modified ZP-Address in AR
0002	0FC3	TEMP TO BUS° AR TO AB° ENH° PC LD°	LB of modified indirect address to bus modified ZP-Address (HB)to AB to read HB of modified indirect address load complete indirect address
next		CLK PC°	into PC
cmd	0FC4	{FETCH READ COMMAND}	
	0FC5	{FETCH DECODE COMMAND}	

HEX	ASSEMBLER INSTR.	CCR
20	JSR \$0208	- - - - -
08		
02		

PC	μCNT	μSTEP	COMMENT
0001	0200	[PC TO AB° ENL°	PC to AB to read subroutine-address-LB via low bus
		[CLK TEMP° CLK PC°	into TEMP increment PC to subroutine-address-HB
0002	0201	[PC TO AB° ENH TEMP TO BUS° AR LD°	PC to AB to read subroutine-address-HB via high bus subroutine-address-LB to low bus prepare loading of AR
		[CLK AR° CLK PC°	load subroutine-address into AR PC to next command
next	0202	[PC TO BUS° ENH° I/O° SP to AB°	PC to bus to write PC-HB via high bus out to SP
command		[R/W°	write PC-HB to SP
next	0203	[SP U/D°	decrement SP
command		[CLK SP°	to next free address
0002	0204	[PC TO BUS° ENL° I/O° SP to AB°	PC to bus to write PC-LB via high bus out to SP
next		[R/W°	write PC-LB to SP
command	0205	[PC TO BUS° AR TO ADD° PC FROM BUS/ADD° PC LD° SP U/D°	PC to Adder Channel A AR to Adder Channel B Adder-Result to PC prepare loading of subroutine-address prepare decrementation of SP
		[CLK PC° CLK SP°	load subroutine-address into PC SP to next free address
Addr.			
Sub-	0226	{FETCH READ COMMAND}	
routine	0227	{FETCH DECODE COMMAND}	

JSR rel. 8-Bit-Offset

22

HEX	ASSEMBLER INSTR.	CCR
22	JSR #0F	- - - - -
0F		

PC	μCNT	μSTEP	COMMENT
0001	0220		
Offset		PC TO AB°	PC to AB
LB		ENL°	to read Offset LB via low bus
		Vectors to 001	HB Offset 00/FF
		AR LD°	prepare loading AR
		CLK AR°	write offset (now 16 bit wide) to AR
		CLK PC°	address next command
next	0221		
command		PC TO BUS°	PC to AB
		ENH°	to write PC-HB via high bus
		I/O°	out to
		SP TO AB°	SP
		R/W°	write PC-HB to SP
next	0222		
command		SP U/D°	prepare decrementation of SP
		CLK SP°	SP to next free address
next	0223		
command		PC TO BUS°	PC to AB
		ENL°	to write PC-LB via low bus
		I/O°	out to
		SP TO AB°	SP
		R/W°	write PC-LB to SP
next	0224		
command		PC TO BUS°	PC to Adder Channel A
		AR TO ADD°	AR to Adder Channel B
		PC FROM BUS/ADD°	Adder-Result to PC
		PC LD°	prepare loading of subroutine-address
		SP U/D°	prepare decrementation of SP
		CLK PC°	load subroutine-address into PC
		CLK SP°	SP to next free address
Addr.	0224	{FETCH READ COMMAND}	
Sub-	0225	{FETCH DECODE COMMAND}	
routine			

JSR rel. 16-Bit-Offset

23

HEX	ASSEMBLER INSTR.	CCR
23	JSR #FF0F	- - - - -
0F		
FF		

PC	PCNT	μSTEP	COMMENT
0001	0230	PC TO AB° ENL°	PC to AB to read Offset LB via low bus
Offset LB		CLK TEMP° CLK PC°	into TEMP increment PC to Offset HB
0002	0231	PC TO AB° ENH° TEMP TO BUS°01 AR LD°	PC to AB to read Offset HB via high bus Offset LB to bus prepare loading AR
Offset HB		CLK AR° CLK PC°	write offset (16 bit wide) to AR increment to next command
next command	0232	PC TO BUS° ENH° I/O° SP TO AB°	PC to AB to write PC-HB via high bus out to SP
		R/W°	write PC-LH to SP
next command	0233	SP U/D°	prepare decrementation of SP
		CLK SP°	SP to next free address
next command	0234	PC TO BUS° ENL° I/O° SP TO AB°	PC to AB to write PC-LB via low bus out to SP
		R/W°	write PC-LB to SP
next command	0235	PC TO BUS° AR TO ADD° PC FROM BUS/ADD° PC LD° SP U/D°	PC to Adder Channel A AR to Adder Channel B Adder-Result to PC prepare loading of subroutine-address prepare decrementation of SP
		CLK PC° CLK SP°	load subroutine-address into PC SP to next free address
Addr.			
Sub.	0236	{FETCH READ COMMAND}	
Rout.	0237	{FETCH DECODE COMMAND}	

JSR (abs.)

43

HEX	ASSEMBLER INSTR.	CCR
43	JSR (\$0308)	- - - - -
08		
03		

PC	PCNT	μSTEP	COMMENT
0001	0430	[PC TO AB° ENL°	PC to AB to read ind. address LB via low bus
		[CLK TEMP° CLK PC°	into TEMP increment PC to ind. address HB
0002	0431	[PC TO AB° ENH° TEMP TO BUS° AR LD°	PC to AB to read ind. Address HB via high bus ind. Address LB to low bus prepare AR-loading
		[CLK AR°	write ind. Address into AR
0002	0432	[AR TO AB° ENL°	AR to AB to read abs. address LB via low bus
		[CLK TEMP° CLK AR°	into TEMP increment AR to abs. address HB
0002	0433	[AR TO AB° ENH° TEMP TO BUS° AR LD°	AR to AB to read abs. Address HB via high bus abs. Address LB to low bus prepare AR-loading
		[CLK AR° CLK PC°	save subroutine-address into AR increment PC to next command
next command	0434	[PC TO BUS° ENH° I/O° SP TO AB°	PC to AB to write PC-HB via high bus out to SP
		[R/W°	write PC-LH to SP
next command	0435	[SP U/D°	prepare decrementation of SP
		[CLK SP°	SP to next free address

JSR (abs.)**continued**

next	02436	[PC TO BUS°	PC to AB
command			ENL°	to write PC-LB via low bus
			I/O°	out to
			SP TO AB°	SP
]		
			R/W°	write PC-LB to SP
next	0437	[PC TO BUS°	PC to Adder Channel A
command			AR TO ADD°	AR to Adder Channel B
			PC FROM BUS/ADD°	Adder-Result to PC
			PC LD°	prepare loading of subroutine-address
			SP U/D°	prepare decrementation of SP
]		
			CLK PC°	load subroutine-address into PC
			CLK SP°	SP to next free address
	0438	{FETCH READ COMMAND}		
	0439	{FETCH DECODE COMMAND}		

JSR (abs.,X)

63

HEX	ASSEMBLER INSTR.	CCR
63	JSR (\$0308,X)	- - - - -
08		
03		

PC	PCNT	μSTEP	COMMENT
0001	0630	PC TO AB° ENL°	PC to AB to read ind. address LB via low bus
LB ind. Address		CLK TEMP° CLK PC°	into TEMP increment PC to ind. address HB
0002	0631	PC TO AB° ENH° TEMP TO BUS° X TO ADD° "00" TO ADDER HB B AR FROM BUS/ADD° AR LD°	PC to AB to read ind. Address HB to Adder HB A ind. Address LB to Adder LB A X-REG TO ADDER LB B Adder HB B to 00 Adder result to AR prepare AR-loading
HB ind. Address		CLK AR°	write ind. Address + X into AR
0002	0632	AR TO AB° ENL°	AR to AB to read abs. address LB via low bus
HB ind. Address		CLK TEMP° CLK AR°	into TEMP increment AR to abs. address HB
0002	0633	AR TO AB° ENH° TEMP TO BUS° AR LD°	AR to AB to read abs. Address HB via high bus abs. Address LB to low bus prepare AR-loading
HB ind. Address		CLK AR° CLK PC°	write Abs. Address into AR increment PC to next command
next command	0634	PC TO BUS° ENH° I/O° SP TO AB°	PC to AB to write PC-HB via high bus out to SP
		R/W°	write PC-LH to SP

JSR (abs.,X)**continued**

next command	0635	[SP U/D°	prepare decrementation of SP
			CLK SP°	SP to next free address
next command	0636	[PC TO BUS°	PC to AB
			ENL°	to write PC-LB via low bus
			I/O°	out to
			SP TO AB°	SP
		[R/W°	write PC-LB to SP
next command	0637	[PC TO BUS°	PC to Adder Channel A
			AR TO ADD°	AR to Adder Channel B
			PC FROM BUS/ADD°	Adder-Result to PC
			PC LD°	prepare loading of subroutine-address
			SP U/D°	prepare decrementation of SP
		[CLK PC°	load subroutine-address into PC
			CLK SP°	SP to next free address
	0638	{FETCH READ COMMAND}		
	0639	{FETCH DECODE COMMAND}		

JSR (ZP)

42

HEX	ASSEMBLER INSTR.	CCR
42	JSR (\$08)	- - - - -
08		

PC	μCNT	μSTEP	COMMENT
0001	0420	PC TO AB° ENL° VECTOR to 000 LD AR°	PC to AB to read ind. ZP-Address LB via low bus 00h to high bus prepare AR-loading
		CLK AR° CLK PC°	into AR increment PC to ind. address HB
0001	0421	AR TO AB° ENL°	AR to AB to read abs. Address LB
		CLK TEMP° CLK AR°	into TEMP increment AR to HB-Address
0001	0422	AR TO AB° ENH° TEMP TO BUS° AR LD°	AR to AB to read abs. Address HB via high bus abs. Address LB to low bus prepare AR-loading
		CLK AR° CLK PC°	write Abs. Address into AR increment PC to next command
next command	0423	PC TO BUS° ENH° I/O° SP TO AB°	PC to AB to write PC-HB via high bus out to SP
		R/W°	write PC-LH to SP
next command	0424	SP U/D°	prepare decrementation of SP
		CLK SP°	SP to next free address
next command	0425	PC TO BUS° ENL° I/O° SP TO AB°	PC to AB to write PC-LB via low bus out to SP
		R/W°	write PC-LB to SP

JSR (ZP)**continued**

next	0426	[PC TO BUS°	PC to Adder Channel A
command			AR TO ADD°	AR to Adder Channel B
			PC FROM BUS/ADD°	Adder-Result to PC
			PC LD°	prepare loading of subroutine-address
			SP U/D°	prepare decrementation of SP
		[CLK PC°	load subroutine-address into PC
			CLK SP°	SP to next free address
	0427	{FETCH READ COMMAND}		
	0428	{FETCH DECODE COMMAND}		

JSR (ZP,X)

62

HEX	ASSEMBLER INSTR.	CCR
62	JSR (\$08,X)	- - - - -
08		
0001	0620	
ZP- Addr.	<div> <div>PC TO AB°</div> <div>ENL°</div> <div>VECTOR to 000</div> <div>X TO ADD°</div> <div>HB-ADD 00°</div> <div>ADDER C/C°</div> <div>AR FROM BUS/ADD°</div> <div>AR LD°</div> </div>	<div> <div>PC to AB</div> <div>to read ZP-Address LB via low bus</div> <div>to Adder LB A</div> <div>00h to Adder HB A</div> <div>X to Adder LB B</div> <div>00 to Adder HB B</div> <div>w/o Adder-Carry</div> <div>Adder-result to AR</div> <div>prepare AR-loading</div> </div>
	<div> <div>CLK AR°</div> </div>	<div> <div>(ZP + X) Address to read from into AR</div> </div>
0001	0621	
LB ind. Address	<div> <div>AR TO AB°</div> <div>ENL°</div> </div>	<div> <div>AR to AB</div> <div>to read abs. Address LB</div> </div>
	<div> <div>CLK TEMP°</div> <div>CLK AR°</div> </div>	<div> <div>into TEMP</div> <div>increment AR to HB-Address</div> </div>
0001	0622	
LB ind. Address	<div> <div>AR TO AB°</div> <div>ENH°</div> <div>TEMP TO BUS°</div> <div>AR LD°</div> </div>	<div> <div>AR to AB</div> <div>to read abs. Address HB via high bus</div> <div>abs. Address LB to low bus</div> <div>prepare AR-loading</div> </div>
	<div> <div>CLK AR°</div> <div>CLK PC°</div> </div>	<div> <div>write Abs. Address into AR</div> <div>increment PC to next command</div> </div>
next	0623	
command	<div> <div>PC TO BUS°</div> <div>ENH°</div> <div>I/O°</div> <div>SP TO AB°</div> </div>	<div> <div>PC to AB</div> <div>to write PC-HB via high bus</div> <div>out to</div> <div>SP</div> </div>
	<div> <div>R/W°</div> </div>	<div> <div>write PC-LH to SP</div> </div>
next	0624	
command	<div> <div>SP U/D°</div> </div>	<div> <div>prepare decrementation of SP</div> </div>
	<div> <div>CLK SP°</div> </div>	<div> <div>SP to next free address</div> </div>
next	0625	
command	<div> <div>PC TO BUS°</div> <div>ENL°</div> <div>I/O°</div> <div>SP TO AB°</div> </div>	<div> <div>PC to AB</div> <div>to write PC-LB via low bus</div> <div>out to</div> <div>SP</div> </div>
	<div> <div>R/W°</div> </div>	<div> <div>write PC-LB to SP</div> </div>

JSR (ZP,X)**continued**

next	0526	[PC TO BUS°	PC to Adder Channel A
command			AR TO ADD°	AR to Adder Channel B
			PC FROM BUS/ADD°	Adder-Result to PC
			PC LD°	prepare loading of subroutine-address
			SP U/D°	prepare decrementation of SP
]		
			CLK PC°	load subroutine-address into PC
			CLK SP°	SP to next free address
	0627	{FETCH READ COMMAND}		
	0628	{FETCH DECODE COMMAND}		

RTS

60

<u>HEX</u>		<u>ASSEMBLER INSTR.</u>	<u>CCR</u>
60		RTS	- - - - -

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0001	0600	-	do nothing, U/D° is set to UP
		CLK SP°	increment SP to LB-return-address
0002	0601	SP TO AB° ENL°	LB return-address via low bus
		CLK TEMP° CLK SP°	to TEMP increment SP to HB-return-address
0002	0602	SP TO AB° ENH° TEMP TO BUS° PC LD°	HB-return-address via high bus LB-return-address prepare PC-loading
		CLK PC°	load return-address to PC
0002	0603	-	do nothing
		CLK PC°	increment PC to next command
0604		{FETCH READ COMMAND}	
0605		{FETCH DECODE COMMAND}	

RTI

40

HEX	ASSEMBLER INSTR.	CCR
40	RTI	Status before Int.
—		

PC	μCNT	μSTEP	COMMENT
0001 next cmd.	0400	[-	
		[CLK SP°	increment SP to saved CCR
0001	0401	[SP TO AB° ENL° MUX CCR to 010	SP to bus to read CCR-byte from stack via low bus CCR-Bits from RTI/SP
		[CLK N° CLK V° CLK B° CLK D° CLK I° CLK Z° CLK C° CLK SP°	restore N-Bit restore V-Bit restore B-Bit restore D-Bit restore I-Bit restore Z-Bit restore C-Bit increment SP to saved PC-address LB
0001	0402	[SP TO AB° ENL°	SP to bus to read saved PC-address LB
		[CLK TEMP° CLK SP°	into TEMP increment SP to saved PC-address HB
0001	0403	[SP TO AB° ENH° TEMP TO BUS° PC LD°	SP to bus to read saved PC-address HB saved PC-address LB prepare PC-loading
		[CLK PC°	load original PC-address into PC
0404	{FETCH READ COMMAND}		
0405	{FETCH DECODE COMMAND}		

RMB ZP

XX

HEX	ASSEMBLER INSTR.	CCR
07	RMB0 \$\$	- - - - -
17	RMB1 \$\$	- - - - -
27	RMB2 \$\$	- - - - -
37	RMB3 \$\$	- - - - -
47	RMB4 \$\$	- - - - -
57	RMB5 \$\$	- - - - -
67	RMB6 \$\$	- - - - -
77	RMB7 \$\$	- - - - -

HEX	ASSEMBLER INSTR.	CCR
17	RMB0 \$02	- - - - -
02		

PC	μCNT	μSTEP	COMMENT
0001 ZP- byte	0xx0	[PC TO AB° ENL° VECTORS to 000 AR LD°	PC to AB to read ZP-Address via low bus 00h to bus HB prepare AR-loading
		[CLK AR° CLK PC°	ZP-Address into AR increment PC to next command
0001	0xx1	[AR TO AB° ENL°	AR to AB to read byte to operate on via low bus
		[CLK ALU-IN°	into ALU-IN
0001	0xx2	[ALU-IN TO ALU-B° RMB° ALU to 11011	ALU-IN to ALU-B set appropriate bit at ALU-A to 0 set ALU to A AND B
		[CLK ALU-OUT°	result into ALU-OUT
0001	0xx3	[AR TO AB° ENL° ALU-OUT TO BUS° I/O°	AR to AB to write byte via low bus from ALU-OUT out
		[R/W°	to memory
0xx4	{FETCH READ COMMAND}		
0xx5	{FETCH DECODE COMMAND}		

SMB ZP

XX

HEX	ASSEMBLER INSTR.	CCR
87	SMB0 \$\$	- - - - -
97	SMB1 \$\$	- - - - -
A7	SMB2 \$\$	- - - - -
B7	SMB3 \$\$	- - - - -
C7	SMB4 \$\$	- - - - -
D7	SMB5 \$\$	- - - - -
E7	SMB6 \$\$	- - - - -
F7	SMB7 \$\$	- - - - -

HEX	ASSEMBLER INSTR.	CCR
87	SMB0 \$02	- - - - -
02		

PC	μCNT	μSTEP	COMMENT
0001 ZP- byte	0xx0	[PC TO AB° ENL° VECTORS to 000 AR LD°]	PC to AB to read ZP-Address via low bus 00h to bus HB prepare AR-loading
		[CLK AR° CLK PC°]	ZP-Address into AR increment PC to next command
0001	0xx1	[AR TO AB° ENL°] [CLK ALU-IN°]	AR to AB to read byte to operate on via low bus into ALU-IN
0001	0xx2	[ALU-IN TO ALU-B° SMB° ALU to 11110] [CLK ALU-OUT°]	ALU-IN to ALU-B set appropriate bit at ALU-A to 1 set ALU to A AND B result into ALU-OUT
0001	0xx3	[AR TO AB° ENL° ALU-OUT TO BUS° I/O°] [R/W°]	AR to AB to write byte via low bus from ALU-OUT out to memory
	0xx4	{FETCH READ COMMAND}	
	0xx5	{FETCH DECODE COMMAND}	

CLC

18

<u>HEX</u>	<u>ASSEMBLER INSTR.</u>	<u>CCR</u>
18	CLC	- - - - - 0

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0001	0180	<div> MUX CCR to 000 FETCH° PC TO AB° </div>	"0" to CCR-Bits indicate Fetch-cycle read next command (F1)
		<div> CLK C° </div>	
0181		{FETCH DECODE COMMAND}	F2

SEC

38

<u>HEX</u>	<u>ASSEMBLER INSTR.</u>	<u>CCR</u>
38	SEC	- - - - - 1

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0001	0380	<div> MUX CCR to 001 FETCH° PC TO AB° </div>	"1" to CCR-Bits indicate Fetch-cycle read next command (F1)
		<div> CLK C° </div>	
0381		{FETCH DECODE COMMAND}	F2

CLD**D8**

<u>HEX</u>	<u>ASSEMBLER INSTR.</u>	<u>CCR</u>
D8	CLD	- - - 0 - - -

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0001	0D80	<div> MUX CCR to 000 FETCH° PC TO AB° </div>	"0" to CCR-Bits indicate Fetch-cycle read next command (F1)
		<div> CLK D° </div>	
0D81		{FETCH DECODE COMMAND} F2	

SED**F8**

<u>HEX</u>	<u>ASSEMBLER INSTR.</u>	<u>CCR</u>
F8	SED	- - - 1 - - -

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0001	0F80	<div> MUX CCR to 001 FETCH° PC TO AB° </div>	"1" to CCR-Bits indicate Fetch-cycle read next command (F1)
		<div> CLK D° </div>	
0F81		{FETCH DECODE COMMAND} F2	

CLI**58**

<u>HEX</u>	<u>ASSEMBLER INSTR.</u>	<u>CCR</u>
58	CLI	- - - - 0 - -

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0001	0580	[MUX CCR to 000 FETCH° PC TO AB°	"0" to CCR-Bits indicate Fetch-cycle read next command (F1)
		[CLK I°	
0581		{FETCH DECODE COMMAND} F2	

SEI**78**

<u>HEX</u>	<u>ASSEMBLER INSTR.</u>	<u>CCR</u>
88	SEI	- - - - 1 - -

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0001	0780	[MUX CCR to 001 FETCH° PC TO AB°	"1" to CCR-Bits indicate Fetch-cycle read next command (F1)
		[CLK I°	
0781		{FETCH DECODE COMMAND} F2	

CLV

B8

HEX	ASSEMBLER INSTR.	CCR
B8	CLV	- 0 - - - - -

PC	μCNT	μSTEP	COMMENT
0001	0B80	[MUX CCR to 000 FETCH° PC TO AB°	"0" to CCR-Bits indicate Fetch-cycle read next command (F1)
] CLK V°	
0B81		{FETCH DECODE COMMAND}	F2

DEA

3A

HEX	ASSEMBLER INSTR.	CCR
3A	DEA	N - - - - Z -

PC	μCNT	μSTEP	COMMENT
0001	0CA0	ACCU TO ALU A° MUX ALU to 01 ALU to 01111 ALU-OUT TO BUS° MUX CCR to 011 FETCH° PC TO AB°	ACCU to ALU A 1 to ALU-C (means without carry = decrement) ALU to A = A - 1 ALU-result to bus CCR from bus indicate Fetch-cycle read next command (F1)
		CLK ACCU° CLK N° CLK Z°	into ACCU
	0CA1	{FETCH DECODE COMMAND} F2	

INA

1A

HEX	ASSEMBLER INSTR.	CCR
1A	INA	N - - - - Z -

PC	μCNT	μSTEP	COMMENT
0001	0E80	ACCU TO ALU A° MUX ALU to 00 ALU to 00000 ALU-OUT TO BUS° MUX CCR to 011 FETCH° PC TO AB°	ACCU to ALU A 0 to ALU-C (means with carry = increment) ALU to A = A + 1 ALU-result to bus CCR from bus indicate Fetch-cycle read next command (F1)
		CLK ACCU° CLK N° CLK Z°	into ACCU
	0E81	{FETCH DECODE COMMAND} F2	

DEX

CA

HEX	ASSEMBLER INSTR.	CCR
CA	DEX	N - - - - Z -

PC	μCNT	μSTEP	COMMENT
0001	0CA0	[
next		X TO ALU A°	X to ALU A
cmd.		MUX ALU to 01	1 to ALU-C (means without carry = decrement)
		ALU to 01111	ALU to A = A - 1
		ALU-OUT TO BUS°	ALU-result to bus
		MUX CCR to 011	CCR from bus
		FETCH°	indicate Fetch-cycle
		PC TO AB°	read next command (F1)
]	
		CLK X°	into X
		CLK N°	
		CLK Z°	
]	
	0CA1	{FETCH DECODE COMMAND}	F2

INX

E8

HEX	ASSEMBLER INSTR.	CCR
E8	INX	N - - - - Z -

PC	μCNT	μSTEP	COMMENT
0001	0E80	[
next		X TO ALU A°	X to ALU A
cmd.		MUX ALU to 00	0 to ALU-C (means with carry = increment)
		ALU to 00000	ALU to A = A + 1
		ALU-OUT TO BUS°	ALU-result to bus
		MUX CCR to 011	CCR from bus
		FETCH°	indicate Fetch-cycle
		PC TO AB°	read next command (F1)
]	
		CLK X°	into X
		CLK N°	
		CLK Z°	
]	
	0E81	{FETCH DECODE COMMAND}	F2

DEY

88

HEX	ASSEMBLER INSTR.	CCR
88	DEY	N - - - - Z -

PC	μCNT	μSTEP	COMMENT
0001	0880	Y TO ALU A°	Y to ALU A
next		MUX ALU to 01	1 to ALU-C (means without carry = decrement)
cmd.		ALU to 01111	ALU to A = A - 1
		ALU-OUT TO BUS°	ALU-result to bus
		MUX CCR to 011	CCR from bus
		FETCH°	indicate Fetch-cycle
		PC TO AB°	read next command (F1)
		CLK Y°	into Y
		CLK N°	
		CLK Z°	
0881		{FETCH DECODE COMMAND} F2	

INY

C8

HEX	ASSEMBLER INSTR.	CCR
C8	INY	N - - - - Z -

PC	μCNT	μSTEP	COMMENT
0001	0C80	Y TO ALU A°	Y to ALU A
next		MUX ALU to 00	0 to ALU-C (means with carry = increment)
cmd.		ALU to 00000	ALU to A = A + 1
		ALU-OUT TO BUS°	ALU-result to bus
		MUX CCR to 011	CCR from bus
		FETCH°	indicate Fetch-cycle
		PC TO AB°	read next command (F1)
		CLK Y°	into Y
		CLK N°	
		CLK Z°	
0C81		{FETCH DECODE COMMAND} F2	

NOP

EA

<hr/>			
<u>HEX</u>	<u>ASSEMBLER INSTR.</u>	<u>CCR</u>	
EA	NOP	- - - - -	
<hr/>			
<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0001	0EA0	{FETCH READ COMMAND}	F1
	0EA1	{FETCH DECODE COMMAND}	F2

PHA

48

HEX	ASSEMBLER INSTR.	CCR
48	PHA	- - - - -

PC	μCNT	μSTEP	COMMENT
0001	0480	<div> <div> ACCU TO BUS° ENL° I/O° SP TO AB° SP U/D° </div> <div> push ACCU via low bus out to SP-address prepare to decrement SP </div> </div>	
		<div> <div>R/W° CLK SP°</div> <div> into SP decrement SP to next free memory-location </div> </div>	
0481		{FETCH READ COMMAND}	F1
0482		{FETCH DECODE COMMAND}	F2

PLA

68

HEX	ASSEMBLER INSTR.	CCR
68	PLA	N - - - - Z -

PC	μCNT	μSTEP	COMMENT
0001	0680	-	do nothing
		CLK SP°	increment SP to last entry
0001	0681	<div> <div>SP TO AB° ENL° MUX CCR to 011</div> <div> SP-Address to AB to read saved ACCU-value via low bus CCR from bus </div> </div>	
		<div> <div>CLK ACCU° CLK N° CLK Z°</div> <div> into ACCU clock in N-Bit of byte read clock in Z-Bit of byte read </div> </div>	
0682		{FETCH READ COMMAND}	F1
0683		{FETCH DECODE COMMAND}	F2

PHP

08

HEX	ASSEMBLER INSTR.	CCR
08	PHP	- - - - -

PC	μCNT	μSTEP	COMMENT
0001	0080	<div>CCR TO BUS°</div> <div>ENL°</div> <div>I/O°</div> <div>SP TO AB°</div> <div>SP U/D°</div>	<div>push CCR-REG</div> <div>via low bus</div> <div>out to</div> <div>SP-address</div> <div>prepare to decrement SP</div>
		<div>R/W°</div> <div>CLK SP°</div>	<div>into SP</div> <div>decrement SP to next free memory-location</div>
0081		{FETCH READ COMMAND}	F1
0082		{FETCH DECODE COMMAND}	F2

PLP

28

HEX	ASSEMBLER INSTR.	CCR
28	PLP	original values

PC	μCNT	μSTEP	COMMENT
0001	0280	-	do nothing
next command		CLK SP°	increment SP to last entry
0001	0281	<div>SP TO AB°</div> <div>ENL°</div> <div>MUX CCR to 010</div>	<div>SP-Address to AB</div> <div>to read saved CCR-values via low bus</div> <div>get original values from SP</div>
		<div>CLK N°</div> <div>CLK V°</div> <div>CLK B°</div> <div>CLK D°</div> <div>CLK I°</div> <div>CLK Z°</div> <div>CLK C°</div>	<div>clock in original N-bit</div> <div>clock in original V-Bit</div> <div>clock in original B-Bit</div> <div>clock in original D-bit</div> <div>clock in original I-Bit</div> <div>clock in original Z-Bit</div> <div>clock in original C-Bit</div>
0282		{FETCH READ COMMAND}	F1
0283		{FETCH DECODE COMMAND}	F2

PHX

DA

HEX	ASSEMBLER INSTR.	CCR
DA	PHX	- - - - -

PC	μCNT	μSTEP	COMMENT
0001	0DA0	<div> X TO BUS° ENL° I/O° SP TO AB° SP U/D° </div>	<div> push X-REG via low bus out to SP-address prepare to decrement SP </div>
		<div> R/W° CLK SP° </div>	<div> into SP decrement SP to next free memory-location </div>
0DA1		{FETCH READ COMMAND}	F1
0DA2		{FETCH DECODE COMMAND}	F2

PLX

FA

HEX	ASSEMBLER INSTR.	CCR
FA	PLX	N - - - - Z -

PC	μCNT	μSTEP	COMMENT
0001	0FA0	-	do nothing
next command		CLK SP°	increment SP to last entry
0001	0FA1	<div> SP TO AB° ENL° MUX CCR to 011 </div>	<div> SP-Address to AB to read saved X-REG-value via low bus CCR from bus </div>
		<div> CLK X° CLK N° CLK Z° </div>	<div> into X-REG clock in N-Bit of byte read clock in Z-Bit of byte read </div>
0FA2		{FETCH READ COMMAND}	F1
0FA3		{FETCH DECODE COMMAND}	F2

PHY

5A

HEX	ASSEMBLER INSTR.	CCR
5A	PHY	- - - - -

PC	μCNT	μSTEP	COMMENT
0001	05A0	<div>Y TO BUS°</div> <div>ENL°</div> <div>I/O°</div> <div>SP TO AB°</div> <div>SP U/D°</div>	<div>push Y-REG</div> <div>via low bus</div> <div>out to</div> <div>SP-address</div> <div>prepare to decrement SP</div>
		<div>R/W°</div> <div>CLK SP°</div>	<div>into SP</div> <div>decrement SP to next free memory-location</div>
05A1		{FETCH READ COMMAND}	F1
05A2		{FETCH DECODE COMMAND}	F2

PLY

7A

HEX	ASSEMBLER INSTR.	CCR
7A	PLY	N - - - - Z -

PC	μCNT	μSTEP	COMMENT
0001	07A0	-	do nothing
		CLK SP°	increment SP to last entry
0001	07A1	<div>SP TO AB°</div> <div>ENL°</div> <div>MUX CCR to 011</div>	<div>SP-Address to AB</div> <div>to read saved Y-REG-value via low bus</div> <div>get N, Z from bus</div>
		<div>CLK Y°</div> <div>CLK N°</div> <div>CLK Z°</div>	<div>into Y-REG</div> <div>clock in N-Bit of byte read</div> <div>clock in Z-Bit of byte read</div>
07A2		{FETCH READ COMMAND}	F1
07A3		{FETCH DECODE COMMAND}	F2

TXA**8A**

<u>HEX</u>	<u>ASSEMBLER INSTR.</u>	<u>CCR</u>
8A	TXA	N - - - - Z -

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0001	08A0	<div> X TO BUS° MUX CCR to 011 FETCH° PC TO AB° </div>	X to bus CCR from bus indicate Fetch-cycle read next command (F1)
		<div> CLK ACCU° CLK N° CLK Z° </div>	into ACCU
08A1		{FETCH DECODE COMMAND} F2	

TAX**AA**

<u>HEX</u>	<u>ASSEMBLER INSTR.</u>	<u>CCR</u>
AA	TAX	N - - - - Z -

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0001	0AA0	<div> ACCU TO BUS° MUX CCR to 011 FETCH° PC TO AB° </div>	ACCU to bus CCR from bus indicate Fetch-cycle read next command (F1)
		<div> CLK X° CLK N° CLK Z° </div>	into X
0AA1		{FETCH DECODE COMMAND} F2	

TXS

9A

HEX	ASSEMBLER INSTR.	CCR
9A	TXS	N - - - - Z -

PC	μCNT	μSTEP	COMMENT
0001	09A0	<div> X TO BUS° MUX CCR to 011 SP LD° FETCH° PC TO AB° </div>	X to bus CCR from bus prepare SP-loading indicate Fetch-cycle read next command (F1)
		<div> CLK SP° CLK N° CLK Z° </div>	into SP
09A1		{FETCH DECODE COMMAND} F2	

TSX

BA

HEX	ASSEMBLER INSTR.	CCR
BA	TSX	N - - - - Z -

PC	μCNT	μSTEP	COMMENT
0001	0BA0	<div> SP TO BUS° MUX CCR to 011 FETCH° PC TO AB° </div>	SP to bus CCR from bus indicate Fetch-cycle read next command (F1)
		<div> CLK X° CLK N° CLK Z° </div>	into X
0BA1		{FETCH DECODE COMMAND} F2	

TYA

98

<u>HEX</u>	<u>ASSEMBLER INSTR.</u>	<u>CCR</u>
98	TYA	N - - - - Z -

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0001	0980	<div> Y TO BUS° MUX CCR to 011 FETCH° PC TO AB° </div>	Y to bus CCR from bus indicate Fetch-cycle read next command (F1)
		<div> CLK ACCU° CLK N° CLK Z° </div>	into ACCU
0981		{FETCH DECODE COMMAND} F2	

TAY

A8

<u>HEX</u>	<u>ASSEMBLER INSTR.</u>	<u>CCR</u>
A8	TAY	N - - - - Z -

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0001	0A80	<div> ACCU TO BUS° MUX CCR to 011 FETCH° PC TO AB° </div>	ACCU to bus CCR from bus indicate Fetch-cycle read next command (F1)
		<div> CLK Y° CLK N° CLK Z° </div>	into Y
0A81		{FETCH DECODE COMMAND} F2	

Befehle nach Adressierungsart

Um den Umfang der Dokumentation nicht ausufern zu lassen, wurden für Befehle, die ein Argument adressieren, das Argument laden / bearbeiten und ggfs. zurückschreiben, jeweils gleichartige Funktionsabschnitte in Tabellen und Sequenz-Darstellungen zusammengefasst.

Ein solcher Befehl setzt sich aus folgenden Teilabschnitten zusammen:

- Adressierung einer Speicherstelle oder eines Registers, aus der/dem das Argument zu laden bzw. bei STORE-Befehlen zu speichern ist. Die konkreten Teilschritte für die Adressierungsarten findet man immer im Anschluss an die Tabelle ADDRESSING-MODE
- Transport des Arguments in ein internes Register (s. Abschnitt DIRECTION: MEMORY TO REGISTER)
- bzw. bei STORE-Befehlen:
Transport eines Registerinhaltes in eine Speicherstelle oder den ACCU (s. Abschnitt DIRECTION: RESULT TO MEMORY/ACCU)
- ggfs. Funktion
Manipulation des Arguments
(Funktion: s. Abschnitt FUNCTIONS "Funktionsname")
- ggfs. abspeichern
Transport des Ergebnisses in eine Speicherstelle oder ACCU (s. Abschnitt DIRECTION: RESULT TO MEMORY/ACCU)

F1 und F2 für das Laden und Dekodieren eines neuen Befehls.

In den Adressierungstabellen sind in der linken Spalte die Befehle aufgeführt, die die jeweilige Adressierungsart unterstützen, gefolgt von der Einsprungadresse des Befehls in das µCode-ROM (Page-Bit und Hex-Wert des Bytes, die 16 möglichen µSteps sind durch X gekennzeichnet). Es schließen sich elf Taktspalten an (Cycles 0 - A), in denen die einzelnen Kurzbezeichnungen für die erforderlichen Teilschritte aufgeführt sind.

Zunächst findet man die entsprechend der Adressierungsart notwendige Anzahl der Adress-Teilschritt-Bezeichnungen (Teilsequenz: „woher kommen die Daten?“), aufsteigend nummeriert (ADDR. x).

Folgende Adressierungs-Sequenzen können gewählt werden:

<u>Adressierungsart</u>	<u>Kurzbezeichnung in Tabelle</u>
ACCU	ACCU TO SHIFT] / SHIFT TO ACCU]
Immediate (Action to ACCU)	PC ALU-IN / PC BIT ALU-IN ²
Immediate (Action to X or Y)	PC ALU-IN / PC BIT ALU-IN ²
Immediate (Load to A or X or Y)	PC ACCU / PC XREG / PC YREG ³
	² Der Programmzähler PC adressiert die gewünschte Speicherstelle, deren Inhalt zur Vorbereitung der ACCU- Aktion in ALU-IN übertragen wird.
	³ Der Programmzähler PC adressiert die gewünschte Speicherstelle, deren Inhalt in das jeweilige Register übertragen wird.
Absolute	Abs
Absolute,X	Abs,X
Absolute,Y	Abs,Y
Zero Page	ZP
Zero Page,X	ZP,X
Zero Page,Y	ZP,Y
(Zero Page)	(ZP)
(Zero Page,X)	(ZP,X)
(Zero Page),Y	(ZP),Y)
Im nächsten Feld steht die Kurzbezeichnung für das Laden des adressierten Arguments in ein internes Zielregister (Teilsequenz: „wohin sollen die Daten“).	

Folgende Richtungs-Sequenzen könne gewählt werden:

Direction (Memory to Register):

Kurzbezeichnung in Tabelle

AR ALU-IN
AR BIT ALU-IN
AR ACCU
AR XREG
AR YREG
AR SHIFT

Dabei adressiert die im Adressregister AR aufbereitete Adresse die gewünschte Speicherstelle, deren Inhalt in das jeweilige Zielregister übertragen wird.

Im nächsten Feld steht, wenn es sich nicht um einen reinen Lade-Befehl handelt, die eigentliche Funktion, die durchgeführt werden soll (Teilsequenz: „was soll mit dem Datenbyte passieren“).

Kurzbezeichnungen für Funktionen

ADC H
ADC D
SBC H
SBC D
AND
ASL
BIT
CMP
CPX
CPY
DEC ACCU
DEC X-REG
DEC Y-REG
EOR
INC ACCU
INC X-REG
INC Y-REG
LSR
ORA
ROL
ROR
TRB
TSB

In dem folgenden Feld ist ggfs. die Kurzbezeichnung für das Rückspeichern des Ergebnisses einer Aktion in ein internes Register oder Speicherstelle aufgeführt (Teilsequenz: „wohin soll das Resultat“)

Kurzbezeichnung für Rückspeichern

ACCU TO MEM
XREG TO MEM
YREG TO MEM
ZERO TO MEM
SHIFT TO MEM
SHIFT TO ACCU

Zum Schluss folgen die beiden Fetch-Cycles F1 und F2 für das Laden und Dekodieren des nächsten Befehls.

Eine Adressierungstabelle mit ihren Abschnittsbezeichnungen stellt eine Liste von Querverweisen dar. Die in den einzelnen Abschnitten aufgeführten Teilsequenzen bilden dann die für den Befehl erforderliche komplette Sequenz.

Beispiel:

Cycle 0 und Cycle 1:

Für den Befehl LSR ABS,X ist zunächst die Tabelle Absolute, X Adressing-Mode zu wählen. Dort findet sich unter LSR in den ersten beiden Feldern mit Abs,X 0 und Abs,X 1 den Querverweis zu den zwei Teilsequenzen für die spezifische Adressierung, die im Detail im Anschluss an die Tabelle zu finden sind (erster und zweiter Teilschritt).

Cycle 2:

Das folgende Feld (dritter Teilschritt) verweist auf die Übersicht „Direction (Memory to Register)“ mit der Teilfunktion AR SHIFT (AR adressiert die Speicherstelle, deren Inhalt in das Schieberegister übertragen wird).

Cycle 3:

Über das nächste Feld (LSR, Teilschritt 4) wird die Verbindung zur Übersicht FUNCTION hergestellt. Dort ist die unter FUNCTION LSR aufgeführten Teil-Sequenz für den Schiebevorgang aufgeführt.

Cycle 4:

In diesem Feld (Teilschritt 5) wird auf die Funktion SHIFT MEM im Abschnitt Direction (Result to Memory/ACCU) verwiesen. Dort findet man die Teilsequenz für das Rückspeichern des geschobenen Wertes in die Speicherstelle.

Cycle 5 und Cycle 6:

Als Abschluss folgen noch die Teilsschritte 6 und 7 mit den Sequenzen für F1 und F2.

ACCU Addressing-Mode

CMD	HEX	CYCLE 0	CYCLE 1	CYCLE 2	CYCLE 3	CYCLE 4	CYCLE 5	CYCLE 6	CYCLE 7	CYCLE 8	CYCLE 9
ASL	0 0A X	ACCU TO SHIFT	ALS	SHIFT TO ACCU, F1	F2	-	-	-	-	-	-
DEA	0 3A X	DEA F1	F2	-	-	-	-	-	-	-	-
INA	0 1A X	INA F1	F2	-	-	-	-	-	-	-	-
LSR	0 4A X	ACCU TO SHIFT	LSR	SHIFT TO ACCU, F1	F2	-	-	-	-	-	-
ROL	0 2A X	ACCU TO SHIFT	ROL	SHIFT TO ACCU, F1	F2	-	-	-	-	-	-
ROR	0 6A X	ACCU TO SHIFT	ROR	SHIFT TO ACCU, F1	F2	-	-	-	-	-	-

ACCU TO SHIFT

PC	<u>μCNT</u> 0XXX	<u>μSTEP</u>	COMMENT
		ACCU TO BUS°	ACCU to bus
		CLK SHIFT°	load SHIFT-Reg.

SHIFT TO ACCU

PC	<u>μCNT</u> 0XXX	<u>μSTEP</u>	COMMENT
		SHIFT-Reg. to 00	Shift-Reg. to bus
		CLK ACCU°	write result to ACCU
		CLK N°	
		CLK Z°	
		CLK C°	

Immediate Addressing-Mode (Action to ACCU)

CMD	HEX	CYCLE 0	CYCLE 1	CYCLE 2	CYCLE 3	CYCLE 4	CYCLE 5	CYCLE 6	CYCLE 7	CYCLE 8	CYCLE 9
ADC	0 69 X	PC ALU-IN	ADC H F1	F2	-	-	-	-	-	-	-
ADC	1 69 X	PC ALU-IN	ADC D	DAA F1	F2	-	-	-	-	-	-
AND	0 29 X	PC ALU-IN	AND F1	F2	-	-	-	-	-	-	-
BIT	0 89 X	PC BIT ALU-IN	BIT F1	F2	-	-	-	-	-	-	-
CMP	0 C9 X	PC ALU-IN	CMP F1	F2	-	-	-	-	-	-	-
EOR	0 49 X	PC ALU-IN	EOR F1	F2	-	-	-	-	-	-	-
ORA	0 09 X	PC ALU-IN	ORA F1	F2	-	-	-	-	-	-	-
SBC	0 E9 X	PC ALU-IN	SBC H F1	F2	-	-	-	-	-	-	-
SBC	1 E9 X	PC ALU-IN	SBC D	DAS F1	F2	-	-	-	-	-	-

Immediate Addressing-Mode (Action to X or Y)

CMD	HEX	CYCLE 0	CYCLE 1	CYCLE 2	CYCLE 3	CYCLE 4	CYCLE 5	CYCLE 6	CYCLE 7	CYCLE 8	CYCLE 9
CPX	0 E0 X	PC ALU-IN	CPX F1	F2	-	-	-	-	-	-	-
CPY	0 C0 X	PC ALU-IN	CPY F1	F2	-	-	-	-	-	-	-

PC ALU-IN / PC BIT ALU-IN

PC	μCNT	μSTEP	COMMENT
0001	0XX0		
Address		PC TO AB°	PC to AB
LB		ENL°	to read immediate value via low bus
		MUX CCR to 011	CCR from bus (BIT-command only)
		CLK ALU-IN°	into ALU Channel B
		CLK PC°	increment PC to next command
		CLK N°	M7 to N-Bit (BIT-command only)
		CLK V°	M6 to N-Bit (BIT-command only)

Immediate Addressing-Mode (Load to A or X or Y)

CMD	HEX	CYCLE 0	CYCLE 1	CYCLE 2	CYCLE 3	CYCLE 4	CYCLE 5	CYCLE 6	CYCLE 7	CYCLE 8	CYCLE A
LDA	0 A9 X	PC ACCU	F1	F2	-	-	-	-	-	-	-
LDX	0 A2 X	PC XREG	F1	F2	-	-	-	-	-	-	-
LDY	0 A0 X	PC YREG	F1	F2	-	-	-	-	-	-	-

PC ACCU

PC	μCNT	μSTEP	COMMENT
0001	0XX0		
Address		MUXCCR to 011	CCR from bus
LB		PC TO AB°	PC to AB
		ENL°	to read immediate value via low bus
		CLK ACCU°	into ACCU
		CLK N°	Clock N-Bit
		CLK Z°	Clock Z-Bit

PC XREG

PC	μCNT	μSTEP	COMMENT
0001	0XX0		
Address		MUX CCR to 011	CCR from bus
LB		PC TO AB°	PC to AB
		ENL°	to read immediate value via low bus
		CLK X°	into XREG
		CLK N°	Clock N-Bit
		CLK Z°	Clock Z-Bit

PC YREG

PC	μCNT	μSTEP	COMMENT
0001	0XX0		
Address		MUX CCR to 011	CCR from bus
LB		PC TO AB°	PC to AB
		ENL°	to read immediate value via low bus
		CLK Y°	into YREG
		CLK N°	Clock N-Bit
		CLK Z°	Clock Z-Bit

Absolute Addressing-Mode

CMD	HEX	CYCLE 0	CYCLE 1	CYCLE 2	CYCLE 3	CYCLE 4	CYCLE 5	CYCLE 6	CYCLE 7	CYCLE 8	CYCLE 9
ADC	0 6D X	Abs. 0	Abs. 1	AR ALU-IN	ADC H F1	F2	-	-	-	-	-
ADC	1 6D X	Abs. 0	Abs. 1	AR ALU-IN	ADC D	DAA F1	F2	-	-	-	-
AND	0 2D X	Abs. 0	Abs. 1	AR ALU-IN	AND F1	F2	-	-	-	-	-
ASL	0 0E X	Abs. 0	Abs. 1	AR SHIFT	ASL	SHIFT TO MEM	F1	F2	-	-	-
BIT	0 2C X	Abs. 0	Abs. 1	AR-BIT ALU-IN	BIT F1	F2	-	-	-	-	-
CMP	0 CD X	Abs. 0	Abs. 1	AR ALU-IN	CMP F1	F2	-	-	-	-	-
CPX	0 EC X	Abs. 0	Abs. 1	AR ALU-IN	CPX F1	F2	-	-	-	-	-
CPY	0 CC X	Abs. 0	Abs. 1	AR ALU-IN	CPY F1	F2	-	-	-	-	-
DEC	0 CE X	Abs. 0	Abs. 1	AR ALU-IN	DEC ALU-IN	F1	F2	-	-	-	-
EOR	0 4D X	Abs. 0	Abs. 1	AR ALU-IN	EOR F1	F2	-	-	-	-	-
INC	0 EE X	Abs. 0	Abs. 1	AR ALU-IN	INC ALU-IN	F1	F2	-	-	-	-
LDA	0 AD X	Abs. 0	Abs. 1	AR ACCU	F1	F2	-	-	-	-	-
LDX	0 AE X	Abs. 0	Abs. 1	AR XREG	F1	F2	-	-	-	-	-
LDY	0 AC X	Abs. 0	Abs. 1	AR YREG	F1	F2	-	-	-	-	-
LSR	0 4E X	Abs. 0	Abs. 1	AR SHIFT	LSR	SHIFT TO MEM	F1	F2	-	-	-
ORA	0 0D X	Abs. 0	Abs. 1	AR ALU-IN	ORA F1	F2	-	-	-	-	-
ROL	0 2E X	Abs. 0	Abs. 1	AR SHIFT	ROL	SHIFT TO MEM	F1	F2	-	-	-
ROR	0 6E X	Abs. 0	Abs. 1	SHIFT	ROR	SHIFT TO MEM	F1	F2	-	-	-
SBC	0 ED X	Abs. 0	Abs. 1	AR ALU-IN	SBC H F1	F2	-	-	-	-	-
SBC	1 ED X	Abs. 0	Abs. 1	AR ALU-IN	SBC D	DAS F1	F2	-	-	-	-
STA	0 8D X	Abs. 0	Abs. 1	ACCU TO MEM	F1	F2	-	-	-	-	-
STX	0 8E X	Abs. 0	Abs. 1	XREG TO MEM	F1	F2	-	-	-	-	-
STY	0 8C X	Abs. 0	Abs. 1	XREG TO MEM	F1	F2	-	-	-	-	-
STZ	0 9C X	Abs. 0	Abs. 1	ZERO TO MEM	F1	F2	-	-	-	-	-
TRB	0 1C X	Abs. 0	Abs. 1	AR ALU-IN	TRB	F1	F2	-	-	-	-
TSB	0 0C X	Abs. 0	Abs. 1	AR ALU-IN	TSB	F1	F2	-	-	-	-

ABS.

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0001	0XX0		
Address		[PC TO AB°	PC to AB
LB		ENL°	to read absolute address LB of command via low bus
		[CLK TEMP°	into TEMP
		CLK PC°	increment PC to HB
0002	0XX1		
Address		[PC TO AB°	PC to AB
HB		ENH°	to read absolute address HB of command via high bus
		TEMP TO BUS°	LB address out
		AR LD°	prepare loading of AR
		[CLK AR°	load AR
		CLK PC°	increment PC to next command

Absolute,X Addressing-Mode

CMD	HEX	CYCLE 0	CYCLE 1	CYCLE 2	CYCLE 3	CYCLE 4	CYCLE 5	CYCLE 6	CYCLE 7	CYCLE 8	ACT. 9
ADC	0 7D X	Abs,X 0	Abs,X 1	AR ALU-IN	ADC H F1	F2	-	-	-	-	-
ADC	1 7D X	Abs,X 0	Abs,X 1	AR ALU-IN	ADC D	DAA F1	F2	-	-	-	-
AND	0 3D X	Abs,X 0	Abs,X 1	AR ALU-IN	AND F1	F2	-	-	-	-	-
ASL	0 1E X	Abs,X 0	Abs,X 1	AR SHIFT	ASL	SHIFT TO MEM	F1	F2	-	-	-
BIT	0 3C X	Abs,X 0	Abs,X 1	AR-BIT ALU-IN	BIT F1	F2	-	-	-	-	-
CMP	0 DD X	Abs,X 0	Abs,X 1	AR ALU-IN	CMP F1	F2	-	-	-	-	-
DEC	0 DE X	Abs,X 0	Abs,X 1	AR ALU-IN	DEC ALU-IN	F1	F2	-	-	-	-
EOR	0 5D X	Abs,X 0	Abs,X 1	AR ALU-IN	EOR F1	F2	-	-	-	-	-
INC	0 FE X	Abs,X 0	Abs,X 1	AR ALU-IN	INC ALU-In	F1	F2	-	-	-	-
LDA	0 BD X	Abs,X 0	Abs,X 1	AR ACCU	F1	F2	-	-	-	-	-
LDY	0 BC X	Abs,X 0	Abs,X 1	AR YREG	F1	F2	-	-	-	-	-
LSR	0 5E X	Abs,X 0	Abs,X 1	AR SHIFT	LSR	SHIFT TO MEM	F1	F2	-	-	-
ORA	0 1D X	Abs,X 0	Abs,X 1	AR ALU-IN	ORA F1	F2	-	-	-	-	-
ROL	0 3E X	Abs,X 0	Abs,X 1	AR SHIFT	ROL	SHIFT TO MEM	F1	F2	-	-	-
ROR	0 7E X	Abs,X 0	Abs,X 1	AR SHIFT	ROR	MEM	F1	F2	-	-	-
SBC	0 FD X	Abs,X 0	Abs,X 1	AR ALU-IN	SBC H F1	F2	-	-	-	-	-
SBC	1 FD X	Abs,X 0	Abs,X 1	AR ALU-IN	SBC D	DAS F1	F2	-	-	-	-
STA	0 9D X	Abs,X 0	Abs,X 1	ACCU TO MEM	F1	F2	-	-	-	-	-
STZ	0 9E X	Abs,X 0	Abs,X 1	ZERO TO MEM	F1	F2	-	-	-	-	-

ABS,X

PC	μCNT	μSTEP	COMMENT
0001	0XX0		
Address		PC TO AB°	PC to AB
LB		ENL°	to read absolute address LB of command via low bus
		CLK TEMP°	into TEMP
		CLK PC°	increment PC to HB
0002	0XX1		
Address		PC TO AB°	PC to AB
HB		ENH°	to read Address-HB to Adder HB A
		TEMP TO BUS°	Address-LB to Adder LB A
		"00" TO ADDER HB B°	00h to Adder HB B
		X TO ADD°	X to Adder LB B
		AR FROM ADD°	Adder-result to AR
		AR LD°	prepare AR-loading
		CLK AR°	ZP,X-Address to read from into AR
		CLK PC°	increment PC to next command

Absolute,Y Addressing-Mode

CMD	HEX	CYCLE 0	CYCLE 1	CYCLE. 2	CYCLE 3	CYCLE 4	CYCLE 5	CYCLE 6	CYCLE 7	CYCLE 8	CYCLE 9
ADC	0 79 X	Abs,Y 0	Abs,Y 1	AR ALU-IN	ADC H F1	F2	-	-	-	-	-
ADC	1 79 X	Abs,Y 0	Abs,Y 1	AR ALU-IN	ADC D	DAA F1	F2	-	-	-	-
AND	0 39 X	Abs,Y 0	Abs,Y 1	AR ALU-IN	AND F1	F2	-	-	-	-	-
CMP	0 D9 X	Abs,Y 0	Abs,Y 1	AR ALU-IN	CMP F1	F2	-	-	-	-	-
EOR	0 59 X	Abs,Y 0	Abs,Y 1	AR ALU-IN	EOR F1	F2	-	-	-	-	-
LDA	0 B9 X	Abs,Y 0	Abs,Y 1	AR ACCU	F1	F2	-	-	-	-	-
LDX	0 BE X	Abs,Y 0	Abs,Y 1	AR YREG	F1	F2	-	-	-	-	-
ORA	0 19 X	Abs,Y 0	Abs,Y 1	AR ALU-IN	ORA F1	F2	-	-	-	-	-
SBC	0 F9 X	Abs,Y 0	Abs,Y 1	AR ALU-IN	SBC H F1	F2	-	-	-	-	-
SBC	1 F9 X	Abs,Y 0	Abs,Y 1	AR ALU-IN	SBC D	DAS F1	F2	-	-	-	-
STA	0 99 X	Abs,Y 0	Abs,Y 1	ACCU TO MEM	F1	F2	-	-	-	-	-

ABS,Y

PC	μCNT	μSTEP	COMMENT
0001	0XX0		
Address		PC TO AB°	PC to AB
LB		ENL°	to read absolute address LB of command via low bus
		CLK TEMP°	into TEMP
		CLK PC°	increment PC to HB
0002	0XX1		
Address		PC TO AB°	PC to AB
HB		ENH°	to read Address-HB to Adder HB A
		TEMP TO BUS°	Address-LB to Adder LB A
		"00" TO ADDER HB B°	00h to Adder HB B
		Y TO ADD°	Y to Adder LB B
		AR FROM ADD°	Adder-result to AR
		AR LD°	prepare AR-loading
		CLK AR°	ZP,Y-Address to read from into AR
		CLK PC°	increment PC to next command

Zero Page Addressing-Mode

CMD	HEX	CYCLE 0	CYCLE 1	CYCLE 2	CYCLE 3	CYCLE 4	CYCLE 5	CYCLE 6	CYCLE 7	CYCLE 8	CYCLE 9
ADC	0 65 X	ZP	AR ALU-IN	ADC H F1	F2	-	-	-	-	-	-
ADC	1 65 X	ZP	AR ALU-IN	ADC D	DAA F1	F2	-	-	-	-	-
AND	0 25 X	ZP	AR ALU-IN	AND F1	F2	-	-	-	-	-	-
ASL	0 06 X	ZP	AR SHIFT	ASL	SHIFT TO MEM	F1	F2	-	-	-	-
BIT	0 24 X	ZP	AR-BIT ALU-IN	BIT F1	F2	-	-	-	-	-	-
CMP	0 C5 X	ZP	AR ALU-IN	CMP F1	F2	-	-	-	-	-	-
CPX	0 E4 X	ZP	AR ALU-IN	CPX F1	F2	-	-	-	-	-	-
CPY	0 C4 X	ZP	AR ALU-IN	CPY F1	F2	-	-	-	-	-	-
DEC	0 C6 X	ZP	AR ALU-IN	DEC ALU-IN	F1	F2	-	-	-	-	-
EOR	0 45 X	ZP	AR ALU-IN	EOR F1	F2	-	-	-	-	-	-
INC	0 E6 X	ZP	AR ALU-IN	INC ALU-IN	F1	F2	-	-	-	-	-
LDA	0 A5 X	ZP	AR ACCU	F1	F2	-	-	-	-	-	-
LDX	0 A6 X	ZP	AR XREG	F1	F2	-	-	-	-	-	-
LDY	0 A4 X	ZP	AR YREG	F1	F2	-	-	-	-	-	-
LSR	0 46 X	ZP	AR SHIFT	LSR	SHIFT TO MEM	F1	F2	-	-	-	-
ORA	0 05 X	ZP	AR ALU-IN	ORA F1	F2	-	-	-	-	-	-
ROL	0 26 X	ZP	AR SHIFT	ROL	SHIFT TO MEM	F1	F2	-	-	-	-
ROR	0 66 X	ZP	SHIFT	ROR	SHIFT TO MEM	F1	F2	-	-	-	-
SBC	0 E5 X	ZP	AR ALU-IN	SBC H F1	F2	-	-	-	-	-	-
SBC	1 E5 X	ZP	AR ALU-IN	SBC D	DAS F1	F2	-	-	-	-	-
STA	0 85 X	ZP	ACCU TO MEM	F1	F2	-	-	-	-	-	-
STX	0 86 X	ZP	XREG TO MEM	F1	F2	-	-	-	-	-	-
STY	0 84 X	ZP	XREG TO MEM	F1	F2	-	-	-	-	-	-
STZ	0 64 X	ZP	ZERO TO MEM	F1	F2	-	-	-	-	-	-
TRB	0 14 X	ZP	AR ALU-IN	TRB	F1	F2	-	-	-	-	-
TSB	0 04 X	ZP	AR ALU-IN	TSB	F1	F2	-	-	-	-	-

ZP

PC	μCNT	μSTEP	COMMENT
0001	0XX0		
ZP-		[PC TO AB°	PC to AB
Byte		ENL°	to read ZP-Address LB via low bus
		VECTORS to 000	00h to bus HB
		AR LD°	prepare AR-loading
		[CLK AR°	ZP-Address to read from into AR
		CLK PC°	increment PC to next command

Zero Page,X Addressing-Mode

CMD	HEX	CYCLE 0	CYCLE 1	CYCLE 2	CYCLE 3	CYCLE 4	CYCLE 5	CYCLE 6	CYCLE 7	CYCLE 8	CYCLE 9
ADC	0 75 X	ZP,X	AR ALU-IN	ADC H F1	F2	-	-	-	-	-	-
ADC	1 75 X	ZP,X	AR ALU-IN	ADC D	DAA F1	F2	-	-	-	-	-
AND	0 35 X	ZP,X	AR ALU-IN	AND F1	F2	-	-	-	-	-	-
ASL	0 16 X	ZP,X	AR SHIFT	ASL	SHIFT TO MEM	F1	F2	-	-	-	-
BIT	0 34 X	ZP,X	AR ALU-IN	BIT F1	F2	-	-	-	-	-	-
CMP	0 D5 X	ZP,X	AR ALU-IN	CMP F1	F2	-	-	-	-	-	-
DEC	0 D6 X	ZP,X	AR ALU-IN	DEC ALU-IN	F1	F2	-	-	-	-	-
EOR	0 55 X	ZP,X	AR ALU-IN	EOR F1	F2	-	-	-	-	-	-
INC	0 F6 X	ZP,X	AR ALU-IN	INC ALU-IN	F1	F2	-	-	-	-	-
LDA	0 B5 X	ZP,X	AR ACCU	F1	F2	-	-	-	-	-	-
LDY	0 B4 X	ZP,X	AR YREG	F1	F2	-	-	-	-	-	-
LSR	0 56 X	ZP,X	AR SHIFT	LSR	SHIFT TO MEM	F1	F2	-	-	-	-
ORA	0 15 X	ZP,X	AR ALU-IN	ORA F1	F2	-	-	-	-	-	-
ROL	0 36 X	ZP,X	AR SHIFT	ROL	SHIFT TO MEM	F1	F2	-	-	-	-
ROR	0 76 X	ZP,X	AR SHIFT	ROR	SHIFT TO MEM	F1	F2	-	-	-	-
SBC	0 F5 X	ZP,X	AR ALU-IN	SBC H F1	F2	-	-	-	-	-	-
SBC	1 F5 X	ZP,X	AR ALU-IN	SBC D	DAS F1	F2	-	-	-	-	-
STA	0 95 X	ZP,X	ACCU TO MEM	F1	F2	-	-	-	-	-	-
STY	0 94 X	ZP,X	YREG TO MEM	F1	F2	-	-	-	-	-	-
STZ	0 74 X	ZP,X	ZERO TO MEM	F1	F2	-	-	-	-	-	-

ZP,X

PC	μCNT	μSTEP	COMMENT
0001	0XX0		
ZP-		PC TO AB°	PC to AB
Address		ENL°	to read ZP-Address via low bus to adder A LB
		X TO ADD°	X to Adder B
		VECTOR to 000	00h to Adder HB A
		"00" TO ADDER HB B°	00h to Adder HB B
		ADDER C/C°	discard carry to HB
		AR FROM ADD°	Adder-result to AR
		AR LD°	prepare AR-loading
		CLK AR°	ZP,X-Address to read from into AR
		CLK PC°	increment PC to next command

Zero Page,Y Addressing-Mode

CMD	HEX	CYCLE 0	CYCLE 1	CYCLE 2	CYCLE 3	CYCLE 4	CYCLE 5	CYCLE 6	CYCLE 7	CYCLE 8	CYCLE 9
LDX	0 B6 X	ZP,Y	AR XREG	F1	F2	-	-	-	-	-	-
STX	0 96 X	ZP,Y	XREG TO MEM	F1	F2	-	-	-	-	-	-

ZP,Y

PC	μCNT	μSTEP	COMMENT
0001	0XX0		
ZP- Address		PC TO AB° ENL° Y TO ADD° VECTOR to 000 "00" TO ADDER HB B° ADDER C/C° AR FROM ADD° AR LD°	PC to AB to read ZP-Address via low bus to adder A LB Y to Adder B 00h to Adder HB A 00h to Adder HB B discard carry to HB Adder-result to AR prepare AR-loading
		CLK AR° CLK PC°	ZP,Y-Address to read from into AR increment PC to next command

(Zero Page) Addressing-Mode

CMD	HEX	CYCLE 0	CYCLE 1	CYCLE 2	CYCLE 3	CYCLE 4	CYCLE 5	CYCLE 6	CYCLE 7	CYCLE 8	CYCLE 9
ADC	0 72 X	(ZP) 0	(ZP) 1	(ZP) 2	AR ALU-IN	ADC H F1	F2	-	-	-	-
ADC	1 72 X	(ZP) 0	(ZP) 1	(ZP) 2	AR ALU-IN	ADC D1	DAA F1	F2	-	-	-
AND	0 32 X	(ZP) 0	(ZP) 1	(ZP) 2	AR ALU-IN	AND F1	F2	-	-	-	-
CMP	0 D2 X	(ZP) 0	(ZP) 1	(ZP) 2	AR ALU-IN	CMP F1	F2	-	-	-	-
EOR	0 52 X	(ZP) 0	(ZP) 1	(ZP) 2	AR ALU-IN	EOR F1	F2	-	-	-	-
LDA	0 B2 X	(ZP) 0	(ZP) 1	(ZP) 2	AR ACCU	F1	F2	-	-	-	-
ORA	0 12 X	(ZP) 0	(ZP) 1	(ZP) 2	AR ALU-IN	ORA F1	F2	-	-	-	-
SBC	0 F2 X	(ZP) 0	(ZP) 1	(ZP) 2	AR ALU-IN	SBC H F1	F2	-	-	-	-
SBC	1 F2 X	(ZP) 0	(ZP) 1	(ZP) 2	AR ALU-IN	SBC D	DAS F1	F2	-	-	-
STA	0 92 X	(ZP) 0	(ZP) 1	(ZP) 2	ACCU TO MEM	F1	F2	-	-	-	-

(ZP)

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0001 LB ind. Address	0XX0	<div>PC TO AB°</div> <div>ENL°</div> <div>VECTOR to 000</div> <div>LD AR°</div>	<div>PC to AB</div> <div>to read ind. ZP-Address LB via low bus</div> <div>00h to high bus</div> <div>prepare AR-loading</div>
		<div>CLK AR°</div> <div>CLK PC°</div>	<div>into AR</div> <div>increment PC to ind. address HB</div>
0001 LB ind. Address	0XX1	<div>AR TO AB°</div> <div>ENL°</div>	<div>AR to AB</div> <div>to read abs. Address LB</div>
		<div>CLK TEMP°</div> <div>CLK AR°</div>	<div>into TEMP</div> <div>increment AR to HB-Address</div>
0001 LB ind. Address	0XX2	<div>AR TO AB°</div> <div>ENH°</div> <div>TEMP TO BUS°</div> <div>AR LD°</div>	<div>AR to AB</div> <div>to read abs. Address HB via high bus</div> <div>abs. Address LB to low bus</div> <div>prepare AR-loading</div>
		<div>CLK AR°</div> <div>CLK PC°</div>	<div>write Abs. Address into AR</div> <div>increment PC to next command</div>

(Zero Page,X) Addressing-Mode

CMD	HEX	CYCLE 0	CYCLE 1	CYCLE 2	CYCLE 3	CYCLE 4	CYCLE 5	CYCLE 6	CYCLE 7	CYCLE 8	CYCLE 9
ADC	0 61 X	(ZP,X) 0	(ZP,X) 1	(ZP,X) 2	AR ALU-IN	ADC H F1	F2	-	-	-	-
ADC	1 61 X	(ZP,X) 0	(ZP,X) 1	(ZP,X) 2	AR ALU-IN	ADC D	DAA F1	F2	-	-	-
AND	0 21X	(ZP,X) 0	(ZP,X) 1	(ZP,X) 2	AR ALU-IN	AND F1	F2	-	-	-	-
CMP	0 C1 X	(ZP,X) 0	(ZP,X) 1	(ZP,X) 2	AR ALU-IN	CMP F1	F2	-	-	-	-
EOR	0 41 X	(ZP,X) 0	(ZP,X) 1	(ZP,X) 2	AR ALU-IN	EOR F1	F2	-	-	-	-
LDA	0 A1 X	(ZP,X) 0	(ZP,X) 1	(ZP,X) 2	AR ACCU	F1	F2	-	-	-	-
ORA	0 01 X	(ZP,X) 0	(ZP,X) 1	(ZP,X) 2	AR ALU-IN	ORA F1	F2	-	-	-	-
SBC	0 E1 X	(ZP,X) 0	(ZP,X) 1	(ZP,X) 2	AR ALU-IN	SBC H F1	F2	-	-	-	-
SBC	1 E1 X	(ZP,X) 0	(ZP,X) 1	(ZP,X) 2	AR ALU-IN	SBC D	DAS F1	F2	-	-	-
STA	0 81 X	(ZP,X) 0	(ZP,X) 1	(ZP,X) 2	ACCU TO MEM	F1	F2	-	-	-	-

(ZP,X)

<u>PC</u>	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
0001 ZP- Addr.	0XX0	<div> <div>PC TO AB° ENL°</div> <div>VECTOR to 000 X TO ADD° HB-ADD 00° ADDER C/C° AR FROM BUS/ADD° AR LD°</div> </div>	<div> <div>PC to AB to read ZP-Address LB via low bus to Adder LB A 00h to Adder HB A X to Adder LB B 00 to Adder HB B w/o Adder-Carry Adder-result to AR prepare AR-loading</div> <div></div> </div>
		<div> <div></div> <div>CLK AR°</div> </div>	<div> <div></div> <div>(ZP + X) Address to read from into AR</div> </div>
0001 LB ind. Address	0XX1	<div> <div>AR TO AB° ENL°</div> <div>CLK TEMP° CLK AR°</div> </div>	<div> <div>AR to AB to read abs. Address LB</div> <div></div> </div>
		<div> <div></div> <div>CLK AR° CLK PC°</div> </div>	<div> <div></div> <div>into TEMP increment AR to HB-Address</div> </div>
0001 LB ind. Address	0XX2	<div> <div>AR TO AB° ENH° TEMP TO BUS° AR LD°</div> <div>CLK AR° CLK PC°</div> </div>	<div> <div>AR to AB to read abs. Address HB via high bus abs. Address LB to low bus prepare AR-loading</div> <div></div> </div>
		<div> <div></div> <div>CLK AR° CLK PC°</div> </div>	<div> <div></div> <div>write Abs. Address into AR increment PC to next command</div> </div>

(Zero Page),Y Addressing-Mode

CMD	HEX	CYCLE 0	CYCLE 1	CYCLE 2	CYCLE 3	CYCLE 4	CYCLE 5	CYCLE 6	CYCLE 7	CYCLE 8	CYCLE 9
ADC	0 71 X	(ZP),Y 0	(ZP),Y 1	(ZP),Y 2	AR ALU-IN	ADC H F1	F2	-	-	-	-
ADC	1 71 X	(ZP),Y 0	(ZP),Y 1	(ZP),Y 2	AR ALU-IN	ADC D	DAA F1	F2	-	-	-
AND	0 31 X	(ZP),Y 0	(ZP),Y 1	(ZP),Y 2	AR ALU-IN	AND F1	F2	-	-	-	-
CMP	0 D1 X	(ZP),Y 0	(ZP),Y 1	(ZP),Y 2	AR ALU-IN	CMP F1	F2	-	-	-	-
EOR	0 51 X	(ZP),Y 0	(ZP),Y 1	(ZP),Y 2	AR ALU-IN	EOR F1	F2	-	-	-	-
LDA	0 B1 X	(ZP),Y 0	(ZP),Y 1	(ZP),Y 2	AR ACCU	F1	F2	-	-	-	-
ORA	0 11 X	(ZP),Y 0	(ZP),Y 1	(ZP),Y 2	AR ALU-IN	ORA F1	F2	-	-	-	-
SBC	0 F1 X	(ZP),Y 0	(ZP),Y 1	(ZP),Y 2	AR ALU-IN	SBC H F1	F2	-	-	-	-
SBC	1 F1 X	(ZP),Y 0	(ZP),Y 1	(ZP),Y 2	AR ALU-IN	SBC D	DAS F1	F2	-	-	-
STA	0 91 X	(ZP),Y 0	(ZP),Y 1	(ZP),Y 2	ACCU TO MEM	F1	F2	-	-	-	-

(ZP),Y

PC	μCNT	μSTEP	COMMENT
0001 ZP- BYTE	0XX0	[PC TO AB° ENL° VECTOR to 000 AR LD°]	PC to AB to read ZP-Address via low bus 00h to bus HB prepare AR-loading
		[CLK AR°]	ZP-Address to read from into AR
0001	0XX1	[AR TO AB° ENL°] [CLK TEMP° CLK AR°]	AR to AB to read low byte of indirect address into TEMP increment ZP-Address in AR
0002 next cmd	0XX3	[AR TO AB° ENH° TEMP TO BUS° "00" TO ADDER HB B° Y TO ADD° AR LD°] [CLK AR° CLK PC°]	AR to AB Ind. Address HB to Adder HB A Ind. Address LB to Adder LB A 00 to Adder HB Channel B Y to Adder LB Channel B load modified indirect address into AR increment PC to next command

Direction (Memory to Register)

AR ALU-IN

PC	<u>μCNT</u> 0XXX	<u>μSTEP</u>	<u>COMMENT</u>
		AR TO AB° ENL°	address of argument out to AB to read byte via low bus
		CLK ALU-IN°	into ALU-IN

AR BIT ALU-IN (BIT-Command immediate only)

PC	<u>μCNT</u> 0XXX	<u>μSTEP</u>	<u>COMMENT</u>
		AR TO AB° ENL° MUX CCR to 011	address of argument out to AB to read byte via low bus CCR from bus
		CLK N° CLK V° CLK ALU-IN°	M7 to N-Bit M6 to N-Bit into ALU-IN

AR ACCU

PC	<u>μCNT</u> 0XXX	<u>μSTEP</u>	<u>COMMENT</u>
		AR TO AB° ENL° MUX CCR to 011	absolute address out to AB to read byte via low bus CCR from bus
		CLK ACCU° CLK N° CLK Z°	into ACCU

AR XREG

PC	<u>μCNT</u> 0XXX	<u>μSTEP</u>	<u>COMMENT</u>
		AR TO AB° ENL° MUX CCR to 011	absolute address out to AB to read byte via low bus CCR from bus
		CLK X° CLK N° CLK Z°	into X-Register

AR YREG

PC	μ CNT	μ STEP	COMMENT
	0XXX		
		AR TO AB ^o ENL ^o MUX CCR to 011	absolute address out to AB to read byte via low bus CCR from bus
		CLK Y ^o CLK N ^o CLK Z ^o	into Y-Register

AR SHIFT

PC	μ CNT	μ STEP	COMMENT
	0XXX		
		AR TO AB ^o ENL ^o SHIFT-Reg. to 11	absolute address out to AB to read byte via low bus Load into register (default)
		CLK SHIFT ^o	into SHIFT-Register

Functions

FUNCTION: ADC H (ADC HEX-MODE)

PC	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
	0XXX	ACCU TO ALU-A° ALU-IN TO ALU-B° MUX ALU to 11 ALU to 01001 ALU TO BUS° MUX CCR to 011	Accu to ALU-A ALU-IN to ALU-B MUX ALU to C° (set carry for addition) ALU to A + B + carry result to bus CCR from bus
		CLK ACCU° CLK N° CLK V° CLK Z° CLK C°	result to Accu

FUNCTION: ADC D (ADC DECIMAL-MODE)

PC	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
	1XXX	ACCU TO ALU-A° ALU-IN TO ALU-B° MUX ALU to 11 ALU to 01001 ALU TO BUS° MUX CCR to 011	Accu to ALU-A ALU-IN to ALU-B MUX ALU to C° (set carry for addition) ALU to A + B + carry ALU to bus CCR from bus
		CLK ACCU° CLK N° CLK V° CLK Z° CLK C°	result to Accu
	1XXX	ACCU TO ALU-A° DAA TO ALU-B° MUX ALU to 01 ALU to 01001 MUX CCR to 011 ALU TO BUS°	Accu to ALU-A FUNCTION: ivate decimal adjust MUX ALU to Carry = 1 (addition without carry) ALU to A + B CCR from bus ALU out to bus
		CLK ACCU° CLK C°	result to Accu

FUNCTION: AND

PC	μ CNT	μ STEP	COMMENT
	0XXX	<div> ALU-IN TO ALU-B° ACCU TO ALU-A° ALU to 11011 MUX CCR to 011 ALU TO BUS° </div>	ALU-IN to ALU-B ACCU to ALU-A ALU to AND CCR from bus result to bus
		<div> CLK N° CLK Z° CLK ACCU° </div>	write result to Accu

FUNCTION: ASL

PC	μ CNT	μ STEP	COMMENT
	0XXX	<div> MUX SHIFT to 0 MUX CCR to 101 SHIFT-Reg. to 01 </div>	0 -> D0 D7 -> Carry Shift left
		<div> CLK SHIFT° CLK C° </div>	shift

FUNCTION: BIT ALU-IN

PC	μ CNT	μ STEP	COMMENT
	0XXX	<div> ALU-IN TO ALU-B° ACCU TO ALU-A° ALU to 11011 MUX CCR to 011 </div>	ALU-IN to ALU-B ACCU to ALU-A ALU to AND CCR from ALU
		<div> CLK Z° </div>	

FUNCTION: CMP (A - M)

PC	μ CNT	μ STEP	COMMENT
	0XXX	<div> ALU-IN TO ALU-B° ACCU TO ALU° MUX ALU to 00 ALU to 00110 MUX CCR to 011 ALU TO BUS° </div>	ALU-IN to ALU-B ACCU to ALU-A 0 to ALU-C (means A - B - 0) ALU to A = A - B CCR from bus result to bus
		<div> CLK N° CLK Z° CLK C° </div>	

FUNCTION: CPX (X - M)

PC	<u>μCNT</u> 0XXX	<u>μSTEP</u>	COMMENT
		ALU-IN TO ALU-B° X TO ALU° MUX ALU to 00 ALU to 00110 MUX CCR to 011 ALU TO BUS°	ALU-IN to ALU-B X to ALU-A 0 to ALU-C (means A - B - 0) ALU to A = A - B CCR from bus result to bus
		CLK N° CLK Z° CLK C°	

FUNCTION: CPY (Y - M)

PC	<u>μCNT</u> 0XXX	<u>μSTEP</u>	COMMENT
		ALU-IN TO ALU-B° Y TO ALU° MUX ALU to 00 ALU to 00110 MUX CCR to 011	ALU-IN to ALU-B Y to ALU-A 0 to ALU-C (means A - B - 0) ALU to A = A - B CCR from bus
		CLK N° CLK Z° CLK C°	

FUNCTION: DEC ALU-IN / MEM

PC	<u>μCNT</u> 0XXX	<u>μSTEP</u>	COMMENT
		ALU-IN TO ALU-A° MUX ALU to 01 ALU to 01111 MUX CCR to 011 ALU TO BUS° AR TO AB° I/O° ENL°	ALU-IN to ALU-A 1 to ALU-Cn° (means without carry = decrement) ALU to A = A - 1 CCR from bus ALU-result to bus address out to AB result out to memory via low bus
		CLK N° CLK Z° R/W°	write result to memory

FUNCTION: DEC ACCU

PC	μ CNT	μ STEP	COMMENT
	0XXX		
		ACCU TO ALU-A°	ACCU to ALU-A
		MUX ALU to 01	1 to ALU-Cn° (means without carry = decrement)
		ALU to 01111	ALU to A = A - 1
		MUX CCR to 011	CCR from bus
		ALU TO BUS°	ALU-result to bus
		CLK N°	
		CLK Z°	
		CLK ACCU°	write result to memory

FUNCTION: DEC X-REG

PC	μ CNT	μ STEP	COMMENT
	0XXX		
		X-REG TO ALU-A°	X-REG to ALU-A
		MUX ALU to 01	1 to ALU-Cn° (means without carry = decrement)
		ALU to 01111	ALU to A = A - 1
		MUX CCR to 011	CCR from bus
		ALU TO BUS°	ALU-result to bus
		CLK N°	
		CLK Z°	
		CLK X-REG°	write result to X-REG

FUNCTION: DEC Y-REG

PC	μ CNT	μ STEP	COMMENT
	0XXX		
		Y-REG TO ALU-A°	Y-REG to ALU-A
		MUX ALU to 01	1 to ALU-Cn° (means without carry = decrement)
		ALU to 01111	ALU to A = A + 1
		MUX CCR to 011	CCR from bus
		ALU TO BUS°	ALU-result to bus
		CLK N°	
		CLK Z°	
		CLK Y-REG°	write result to Y-REG

FUNCTION: EOR

PC	<u>μCNT</u>	<u>μSTEP</u>	COMMENT
	0XXX	ALU-IN TO ALU-B° ACCU TO ALU-A° ALU to 10110 MUX CCR to 011 ALU TO BUS°	ALU-IN to ALU-B ACCU to ALU-A ALU to EOR CCR from bus result to bus
		CLK N° CLK Z° CLK ACCU°	write result to memory

FUNCTION: INC ALU-IN

PC	<u>μCNT</u>	<u>μSTEP</u>	COMMENT
	0XXX	ALU-IN TO ALU-A° MUX ALU to 00 ALU to 00000 MUX CCR to 011 ALU TO BUS° AR TO AB° I/O° ENL°	ALU-IN to ALU-A 0 to ALU-Cn° (means with carry = increment) ALU to A = A + 1 CCR from bus ALU-result to bus address out to AB result out to memory via low bus
		CLK N° CLK Z° R/W°	write result to memory

FUNCTION: INA

PC	<u>μCNT</u>	<u>μSTEP</u>	COMMENT
	0XXX	ACCU TO ALU-A° MUX ALU to 00 ALU to 00000 MUX CCR to 011 ALU TO BUS°	ACCU to ALU-A 0 to ALU-Cn° (means with carry = increment) ALU to A = A + 1 CCR from bus ALU-result to bus
		CLK N° CLK Z° CLK ACCU°	write result to memory

FUNCTION: INC X-REG

PC	<u>μCNT</u>	<u>μSTEP</u>	COMMENT
	0XXX	<div> X-REG TO BUS° MUX ALU to 00 ALU to 00000 MUX CCR to 011 ALU TO BUS° </div>	X-REG to ALU-A 0 to ALU-Cn° (means with carry = increment) ALU to A = A + 1 CCR from bus ALU-result to bus
		<div> CLK N° CLK Z° CLK X-REG° </div>	write result to X-REG

FUNCTION: INC Y-REG

PC	<u>μCNT</u>	<u>μSTEP</u>	COMMENT
	0XXX	<div> Y-REG TO BUS° MUX ALU to 00 ALU to 00000 MUX CCR to 011 ALU TO BUS° </div>	Y-REG to ALU-A 0 to ALU-Cn° (means with carry = increment) ALU to A = A + 1 CCR from bus ALU-result to bus
		<div> CLK N° CLK Z° CLK Y-REG° </div>	write result to Y-REG

FUNCTION: LSR

PC	<u>μCNT</u>	<u>μSTEP</u>	COMMENT
	0XXX	<div> MUX SHIFT to 0 MUX CCR to 100 SHIFT-Reg. to 01 </div>	0 -> D7 D0 -> Carry Shift right
		<div> CLK SHIFT° CLK C° </div>	shift

FUNCTION: ORA

PC	μ CNT	μ STEP	COMMENT
	0XXX	<div> ALU-IN TO ALU-B° ACCU TO ALU-A° ALU to 11110 MUX CCR to 011 ALU TO BUS° </div>	ALU-IN to ALU-B ACCU to ALU-A ALU to OR CCR from bus result to bus
		<div> CLK N° CLK Z° CLK ACCU° </div>	write result to memory

FUNCTION: ROL

PC	μ CNT	μ STEP	COMMENT
	0XXX	<div> MUX SHIFT to 1 MUX CCR to 101 SHIFT-Reg. to 01 </div>	Carry -> D0 D7 -> Carry Shift left
		<div> CLK SHIFT° CLK C° </div>	shift

FUNCTION: ROR

PC	μ CNT	μ STEP	COMMENT
	0XXX	<div> MUX SHIFT to 1 MUX CCR to 100 SHIFT-Reg. to 10 </div>	Carry -> D7 D0 -> Carry Shift right
		<div> CLK SHIFT° CLK C° </div>	shift

FUNCTION: SBC (HEX-MODE): SBC H

PC	μ CNT	μ STEP	COMMENT
	0XXX	<div> ACCU TO ALU-A° ALU-IN TO ALU-B° MUX ALU to 10 ALU to 00110 ALU TO BUS° MUX CCR to 011 </div>	Accu to ALU-A ALU-IN to ALU-B set to C (set to 11 if A - B - C° is required) ALU to A + B + carry result to bus CCR from bus
		<div> CLK ACCU° CLK N° CLK Z° CLK C° </div>	result to Accu

FUNCTION: SBC (DECIMAL-MODE): SBC H & DAS

PC	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
	1XXX	ACCU TO ALU-A° ALU-IN TO ALU-B° MUX ALU to 10 ALU to 01001 ALU TO BUS° MUX CCR to 011	Accu to ALU-A ALU-IN to ALU-B set to C (set to 11 if A - B - C° is required) ALU to A - B - carry ALU to bus CCR from bus
		CLK ACCU° CLK N° CLK Z° CLK C°	result to Accu
	1XXX	ACCU TO ALU-A° DAA TO ALU-B° MUX ALU to 01 ALU to 01001 ALU TO BUS°	Accu to ALU-A FUNCTION: ivate decimal adjust MUX ALU to Carry = 1 (addition without carry) ALU to A + B ALU out to bus
		CLK ACCU°	result to Accu

FUNCTION: TRB

PC	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
	0XXX	ALU-IN TO ALU-B° ACCU TO ALU-A° ALU to 10010 MUX CCR to 011 ALU TO BUS° AR TO AB°	ALU-IN to ALU-B ACCU TO ALU-A ALU to A° AND B CCR from bus result to bus back to Mem
		R/W° CLK Z°	write result

FUNCTION: TSB

PC	<u>μCNT</u>	<u>μSTEP</u>	<u>COMMENT</u>
	0XXX	ALU-IN TO ALU-B° ACCU TO ALU-A° ALU to 11110 MUX CCR to 011 ALU TO BUS° AR TO AB°	ALU-IN to ALU-B ACCU TO ALU-A ALU to A OR B CCR from bus result to bus back to Mem
		R/W° CLK Z°	write result

Direction (Result to Memory/Accu)

ACCU TO MEM

PC	<u>μCNT</u> 0XXX	<u>μSTEP</u>	<u>COMMENT</u>
		<div> <div>AR TO AB°</div> <div>ACCU TO BUS°</div> <div>ENL°</div> <div>I/O°</div> </div>	<div>address out to AB</div> <div>Accu to bus</div> <div>to write byte via low bus</div> <div>out</div>
		<div>R/W°</div>	to memory

XREG TO MEM

PC	<u>μCNT</u> 0XXX	<u>μSTEP</u>	<u>COMMENT</u>
		<div> <div>AR TO AB°</div> <div>X TO BUS°</div> <div>ENL°</div> <div>I/O°</div> </div>	<div>address out to AB</div> <div>X-Reg. to bus</div> <div>to write byte via low bus</div> <div>out</div>
		<div>R/W°</div>	to memory

YREG TO MEM

PC	<u>μCNT</u> 0XXX	<u>μSTEP</u>	<u>COMMENT</u>
		<div> <div>AR TO AB°</div> <div>Y TO BUS°</div> <div>ENL°</div> <div>I/O°</div> </div>	<div>address out to AB</div> <div>Y-Reg. to bus</div> <div>to write byte via low bus</div> <div>out</div>
		<div>R/W°</div>	to memory

ZERO TO MEM

PC	<u>μCNT</u> 0XXX	<u>μSTEP</u>	<u>COMMENT</u>
		<div> <div>AR TO AB°</div> <div>VECTORS to 000</div> <div>ENH°</div> <div>I/O°</div> </div>	<div>address out to AB</div> <div>Vectors to ZP (00h to HB)</div> <div>to write byte via high bus</div> <div>out</div>
		<div>R/W°</div>	to memory

SHIFT TO MEM

PC	<u>μCNT</u> 0XXX	<u>μSTEP</u>	<u>COMMENT</u>
		<div> AR TO AB° SHIFT-Reg to 00 MUX CCR to 011 ENL° I/O° </div>	address out to AB Shift-Reg. to output-mode CCR from BUS/ALU to write byte via low bus out
		<div> R/W° CLK N° CLK Z° </div>	to memory

SHIFT TO ACCU

PC	<u>μCNT</u> 0XXX	<u>μSTEP</u>	<u>COMMENT</u>
		<div> SHIFT-Reg to 00 MUX CCR to 011 </div>	SHIFT-Reg to output-mode CCR from BUS/ALU
		<div> CLK ACCU° CLK N° CLK Z° </div>	into ACCU

Befehlsmatrix

Erläuterung zum Befehlsfeldaufbau:

Befehls-Mnemonic

Adressierungsart
Anzahl der Befehlsbytes
Takte N für 65C02 (*)
Takte N für 65C02+ (*)

(*)

Für Effizienzvergleich Taktangabe des 65C02 mit Faktor 2 multiplizieren (s. Ein-Phasen-Takt)

Erläuterung zu den Grauabstufungen in den Befehlsfeldern:

Befehle des 6502

Befehle des
6502 und
65C02

Befehle des
6502 und
65C02 und
65C02+

Ergänzende Erläuterungen zu den Taktangaben:

- ° 1 Takt zusätzlich für alle ADC/SBC-Befehle im Dezimal-Modus
- ' 1 Takt zusätzlich bei Seitenüberschreitung
- 2 1 Takt zusätzlich bei Sprung in gleicher Seite bzw.
2 Takte zusätzlich bei Sprung in eine andere Seite
- 3 1 Takt zusätzlich bei Sprung

\LN HN \	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BRK IMP 1 7 6 9	ORA (ZP,X) 2 5 6	SWI ZP 8 2 - 9		TSB ZP 2 5 5	ORA ZP 2 3 4	ASL ZP 2 5 6	RMB0 ZP 8 2 5 6	PHP IMP 1 3 3	ORA IMM 2 3 3	ASL ACCU 1 2 4	BBR0 ZP 16 4 - 7	TSB ABS 3 6 6	ORA ABS 3 4 5	ASL ABS 3 7 7	BBR0 ZP 8 3 6 6 ³
1	BPL REL 8 2 2 ² 3 ³	ORA (ZP),Y 2 5' 6	ORA (ZP) 2 5 6	BPL REL 16 3 - 3 ³	TRB ZP 2 5 5	ORA ZP,X 2 4 4	ASL ZP,X 2 6 6	RMB1 ZP 8 2 5 6	CLC IMP 1 2 5	ORA ABS,Y 3 4' 5	INC ACCU 1 2 2	BBR1 ZP 16 4 - 7	TRB ABS 3 6 5	ORA ABS,X 3 4' 5	ASL ASL,X 3 7 7	BBR1 ZP 8 3 5 ² 6 ³
2	JSR ABS 3 6 7	AND (ZP,X) 2 6 6	JSR REL 8 2 - 7	JSR REL 16 2 - 10	BIT ZP 2 3 4	AND ZP 2 3 4	ROL ZP 2 5 6	RMB2 ZP 8 2 5 6	PLP IMP 1 2 4	AND IMM 2 2 3	ROL ACCU 1 2 4	BBR2 ZP 16 4 - 7	BIT ABS 3 4 5	AND ABS 3 4 5	ROL ABS 3 6 7	BBR2 ZP 8 3 5 ² 6 ³
3	BMI REL 8 2 2 ² 3 ³	AND (ZP),Y 2 5' 6	AND (ZP) 2 5 6	BMI REL 16 3 - 3 ³	BIT ZP,X 2 3 4	AND ZP,X 2 4 4	ROL ZP,X 2 6 6	RMB3 ZP 8 2 5 6	SEC IMP 1 2 2	AND ABS,Y 3 4' 5	DEC ACCU 1 2 2	BBR3 ZP 16 4 - 7	BIT ABS,X 3 6 5	AND ABS,X 3 4' 5	ROL ABS,X 3 7 7	BBR3 ZP 8 3 5 ² 6 ³
4	RTI IMP 1 6 6	EOR (ZP,X) 2 6 6	JSR (ZP) 2 - 9	JSR (ABS) 2 - 10		EOR ZP 2 3 4	LSR ZP 2 5 6	RMB4 ZP 8 2 5 6	PHA IMP 1 3 3	EOR IMM 2 2 3	LSR ACCU 1 2 4	BBR4 ZP 16 4 - 7	JMP ABS 3 4 5	EOR ABS 3 4 5	LSR ABS 3 6 7	BBR4 ZP 8 3 5 ² 6 ³
5	BVC REL 8 2 2 ² 3 ³	EOR (ZP),Y 2 5' 6	EOR (ZP) 2 5 6	BVC REL 16 3 - 3 ³		EOR ZP,X 2 4 4	LSR ZP,X 2 6 6	RMB5 ZP 8 2 5 6	CLI IMP 1 2 2	EOR ABS,Y 3 4' 5	PHY IMP 1 3 3	BBR5 ZP 16 4 - 7		EOR ABS,X 3 4' 5	LSR ABS,X 3 7 7	BBR5 ZP 8 3 5 ² 6 ³
6	RTS IMP 1 6 6	ADC (ZP,X) 2 6 ⁰ 6 ⁰	JSR (ZP,X) 2 - 9	JSR (ABS,X) 2 - 10	STZ ZP 2 3 4	ADC ZP 2 3 ⁰ 6	ROR ZP 2 5 6	RMB6 ZP 8 2 5 6	PLA IMP 1 2 4	ADC IMM 2 2 ⁰ 3 ⁰	ROR ACCU 1 3 4	BBR6 ZP 16 4 - 7	JMP (ABS) 3 6 5	ADC ABS 3 4 5	ROR ABS 3 6 7	BBR6 ZP 8 3 5 ² 6 ³
7	BVS REL 8 2 2 ² 3 ³	ADC (ZP),Y 2 5 ⁰ 6 ⁰	ADC (ZP) 2 5 ⁰ 6 ⁰	BVS REL 16 3 - 3 ³	STZ ZP,X 2 4 4	ADC ZP,X 2 4 ⁰ 4 ⁰	ROR ZP,X 2 6 6	RMB7 ZP 8 2 5 6	SEI IMP 1 2 2	ADC ABS,Y 3 4 ⁰ 5 ⁰	PLY IMP 1 4 4	BBR7 ZP 16 4 - 7	JMP (ABS,X) 3 6 6	ADC ABS,X 3 4 ⁰ 5 ⁰	ROR ABS,X 3 7 7	BBR7 ZP 8 3 5 ² 6 ³
8	BRA REL 8 2 3' 4	STA (ZP,X) 2 6 6		BRA REL 16 2 - 5	STY ZP 2 3 4	STA ZP 2 3 4	STX ZP 2 3 4	SMB0 ZP 8 2 5 6	DEY IMP 1 2 2	BIT IMM 2 2 3	TXA IMP 1 2 2	BBS0 ZP 16 4 - 7	STY ABS 3 4 5	STA ABS 3 4 5	STX ABS 3 4 5	BBS0 ZP 8 3 5 ² 6 ³
9	BCC REL 8 2 2 ² 3 ³	STA (ZP),Y 2 6 6	STA (ZP) 2 6 6	BCC REL 16 3 - 3 ³	STY ZP,X 2 4 4	STA ZP,X 2 4 4	STX ZP,Y 2 4 4	SMB1 ZP 8 2 5 6	TYA IMP 1 2 2	STA ABS,Y 3 5 5	TXS IMP 1 2 2	BBS1 ZP 16 4 - 7	STZ ABS 3 4 5	STA ABS,X 3 5 5	STZ ABS,X 3 5 5	BBS1 ZP 8 3 5 ² 6 ³
A	LDY IMP 2 2 3	LDA (ZP,X) 2 6 6	LDX IMM 2 2 3		LDY ZP 2 3 4	LDA ZP 2 3 4	LDX ZP 2 3 4	SMB2 ZP 8 2 5 6	TAY IMP 1 2 2	LDA IMM 2 2 3	TAX IMP 1 2 2	BBS2 ZP 16 4 - 7	LDY ABS 3 4 5	LDA ABS 3 4 5	LDX ABS 3 4 5	BBS2 ZP 8 3 5 ² 6 ³
B	BCS REL 8 2 2 ² 3 ³	LDA (ZP),Y 2 5 6	LDA (ZP) 2 5 6	BCS REL 16 3 - 3 ³	LDY ZP,X 2 4 4	LDA ZP,X 2 4 4	LDX ZP,Y 2 4 4	SMB3 ZP 8 2 5 6	CLV IMP 1 2 2	LDA ABS,Y 3 4' 5	TSX IMP 1 2 2	BBS3 ZP 16 4 - 7	LDY ABS,X 3 4' 5	LDA ABS,X 3 4' 5	LDX ABS,Y 3 4' 5	BBS3 ZP 8 3 5 ² 6 ³
C	CPY IMM 2 2 3	CMP (ZP,X) 2 6 6			CPY ZP 2 3 4	CMP ZP 2 3 4	DEC ZP 2 5 6	SMB4 ZP 8 2 5 6	INY IMP 1 2 2	CMP IMM 2 2 3	DEX IMP 1 2 2	BBS4 ZP 16 4 - 7	CPY ABS 3 4 5	CMP ABS 3 4 5	DEC ABS 3 6 6	BBS4 ZP 8 3 5 ² 6 ³
D	BNE REL 8 2 2 ² 3 ³	CMP (ZP),Y 2 5 6	CMP (ZP) 2 5 7	BNE REL 16 3 - 3 ³		CMP ZP,X 2 4 4	DEC ZP,X 2 6 5	SMB5 ZP 8 2 5 6	CLD IMP 1 2 2	CMP ABS,Y 3 4' 5	PHX IMP 1 3 3	BBS5 ZP 16 4 - 7	JMP (ZP) 2 - 7	CMP ABS,X 3 4 5	DEC ABS,X 3 7 6	BBS5 ZP 8 3 5 ² 6 ³
E	CPX IMM 2 2 3	SBC (ZP,X) 2 6 ⁰ 6 ⁰			CPX ZP 2 3 4	SBC ZP 2 3 ⁰ 4 ⁰	INC ZP 2 5 5	SMB6 ZP 8 2 5 6	INX IMP 1 2 2	SBC IMM 2 2 ⁰ 3 ⁰	NOP IMP 1 2 2	BBS6 ZP 16 4 - 7	CPX ABS 3 4 5	SBC ABS 3 4 ⁰ 5 ⁰	INC ABS 3 6 6	BBS6 ZP 8 3 5 ² 6 ³
F	BEQ REL 8 2 2 ² 3 ³	SBC (ZP),Y 2 5 ⁰ 6 ⁰	SBC (ZP) 2 5 ⁰ 6 ⁰	BEQ REL 16 3 - 3 ³		SBC ZP,X 2 4 ⁰ 4 ⁰	INC ZP,X 2 6 5	SMB7 ZP 8 2 5 6	SED IMP 1 2 2	SBC ABS,Y 3 4 ⁰ 5 ⁰	PLX IMP 1 2 4	BBS7 ZP 16 4 - 7	JMP (ZP,X) 2 - 7	SBC ABS,X 3 4 ⁰ 5 ⁰	INC ABS,X 3 7 6	BBS7 ZP 8 3 5 ² 6 ³

Aussichten

Zurzeit existiert das Projekt mit zusätzlichem Micro-Code-Editor und stufenweise schaltbarem Wait-State-Generator.

Als nächstes Projekt ist die Realisierung einer Ablaufsteuerung mit Zwei-Phasen-Takt und einer Control-Unit ohne Micro-Code-ROM sowie die Entwicklung eines 65C02-Makro-Assemblers, der die Befehle des 65C02+ unterstützt, geplant.

Quellenangaben

Junior-Computer Band 1
A. Nachtmann
G.H. Nachbar
2. Auflage 1980
Elektor Verlag

Struktur und Betrieb digitaler Rechner
Vorlesungsskript
Dipl.-Ing. Klaus Blecken
Fachhochschule Darmstadt
Wintersemester 1979/80

Digitaltechnik I + II
Vorlesungsskript
Dr. Ing. August Reiner
Fachhochschule Darmstadt
Wintersemester 1979/80
Sommersemester 1980

Halbleiterschaltungstechnik
Dr.-Ing. Ulrich Tietze/
Dr.-Ing. Christoph Schenk
6. Auflage
1983

High-Speed CMOS Logic Data Book
Texas Instruments 1984

div. Internetseiten mit Bauteilbeschreibungen
der Firma Texas Instruments (ti.com)