Übersicht über die Befehle des C517

Bedeutung der Abkürzungen in den Befehlen:

Operand	Bedeutung
A	Akkumulator
addrl1	Adresse in einem 2K- Programmspeicherblock
addr16	Adresse i m 64K- Prog ram mspeicherbereich
B	RegisterB
bit	Bitadresse im internen RAM
/bit	Komplementierter Inhalt der Bitadresse
C	Carry
#data	8-Bit-Konstante
#datal6	16-Bit-Konstante
direct	Adresse eines internen RAM-Platzes
DPTR	Datenpointer
PC	Programmzähler
@Ri	Indirekte Adressierung eines RAM-Platzes über R1 oder Ri
Rn	Register RO bis R7 der aktuellen Registerbank
@	Zeichen für indirekte Adressierung
kursiv	Logische Operation
rel	Signiertes 8-Bit-Offset für Sprungbefehle

Bitbefehle:

Mnemonik	Beschreibung	Byte	Maschinen-zyklen
CLR C	Lösche Carry	1	1
CLR bit	Lösche Bit	2	1
SETB C	Setze Carry	1	1
SETB bit	Setze Bit	2	1
CPL C	Komplementiere Carry	1	1
CPL bit	Komplementiere Bit	2	1
ANL C,bit	Carry und Bit	2	2
ANL C,/bit	Carry und nicht Bit	2	2
ORL C,bit	Carry oder Bit	2	2
ORL C,/bit	Carry oder nicht Bit	2	2
MOV C,bit	Schreibe Bit ins Carry	2	1
MOV bit,C	Schreibe Carry ins Bit	2	2

Arithmetikbefehle:

ADD A,Rn	Addiere Register zu Akkumulator	1	1
ADD A,direct	Addiere Direkt-Byte zu Akkumulator	2	1
ADD A,@Ri	Addiere Indirekt-Byte zu Akkumulator	1	1
ADD A,#data	Addiere Konstante zu Akkumulator	2	1
ADDCA,Rn	Addiere Register zu Akkumulator mit Carry	1	1
ADDC A,direct	Addiere Direkt-Byte zu Akkumulator mit Carry	2	1
ADDCA,@Ri	Addiere Indirekt-Byte zu Akkumulator mit Carry	1	1
ADDC A,#data	Addiere Konstante zu Akkumulator mit Carry	2	1
SUBBA,Rn	Subtrahiere Register zu Akkumulator	1	1
SUBB A,direct	Subtrahiere Direkt-Byte zu Akkumulator	2	1
SUBBA,@Ri	Subtrahiere Indirekt-Byte zu Akkumulator	1	1
SUBB A,#data	Subtrahiere Konstante zu Akkumulator	2	1
INC A	Akkumulator + 1	1	1
INC Rn	Rn+1	1	1
INC direct	Direkt-Byte+1	2	1
INC @Ri	Indirekt-Byte+1	1	1
DEC A	Akkumulator-l	1	1
DEC Rn	Rn -1	1	1
DEC direct	Direkt-Byte - 1	2	1
DEC @Ri	Indirekt-Byte-l	1	1
INC DPTR	Datenpointer+1	1	2
MUL AB	Multipliziere A mit B	1	1
DIV AB	Teile A durch B	1	1
DA A	Dezimalkorrektur	1	1

Transferbefehle:

MOV A,Rn	Schreibe Register in den Akkumulator	1	1
MOV A,direct	Schreibe Direkt-Byte in den Akkumulator	2	1
MOV A,@Ri	Schreibe Indirekt-Byte in den Akkumulator	1	1
MOV A,#data	Schreibe Konstante in den Akkumulator	2	1
MOV Rn,A	Schreibe Akkumulator in das Register	1	2
MOV Rn,direct	Schreibe Direkt-Byte in das Register	2	2
MOV Rn,#data	Schreibe Konstante in das Register	2	1
MOV direct,A	Schreibe Akkumulator in das Direkt-Byte	2	1
MOV direct,Rn	Schreibe Register in das Direkt-Byte	2	2
MOV direct,direct	Schreibe Direkt-Byte in das Direkt-Byte	3	2
MOV direct,@Ri	Schreibe Indirekt-Byte in das Direkt-Byte	2	2
MOV direct,#data	Schreibe Konstante in das Direkt-Byte	3	2
MOV @Ri,A	Schreibe Akkumulator in das indirekte Byte	1	1
MOV @Ri,direct	Schreibe Direkt-Byte in das Indirekt-Byte	2	2
MOV @Ri,#data	Schreibe Konstante in das Indirekt-Byte	2	1
MOV DPTR,#datalG	Schreibe 16-Bit-Konstante in den Datenpointer	3	2
MOVCA,@A+DPTR	Schreibe Tabellenkonstante relativ zum Datenpointer in den Akkumulator	1	2
MOVC A,@A+PC	Schreibe Tabellenkonstante relativ zum Programmzähler in den Akku	1	2

Transferbefehle, CALL- und Sprungbefehle-1

MOVXA,@Ri	Schreibe externes RAM-Byte in den Akku	1	2
MOVX A,@DPTR	Schreibe externes RAM-Byte in den Akku	1	2
MOVX@Ri,A	Schreibe Akkumulator in externes RAM	1	2
MOVX @DPTR,A	Schreibe Akkumulator in externes RAM	1	2
PUSH direct	Lege Direkt-Byte auf Stack ab	2	2
POP direct	Schreibe vom Stack in Direkt-Byte	2	2
XCH A,Rn	Tausche Register mit Akkumulator aus	1	1
XCH A,direct	Tausche Direkt-Byte mit Akkumulator aus	2	1
XCH A,@Ri	Tausche Indirekt-Byte mit dem Akku aus	1	1
XCHDA,@Ri	Tausche Low-Nibble indirekt mit Akku aus	1	1

ACALL addrll	Subroutinenaufruf 2K-Block	2	2
LCALL addrl6	Subroutinenaufruf in 64K	3	2
RET	Subroutine beenden	1	2
RETI	Interrupt beenden	1	2
AJMP addrl 1	Sprung im 2K-81ock	2	2
LJMP addrl6	Sprung in 64K	3	2
SJMP rel	relativer Sprung	2	2
JMP @A+DPTR	indirekter Sprung relativ zum Daten-pointer	1	2
JZ rel	Sprung, wenn Akkuinhalt Null ist	2	2
JNZ rel	Sprung, wenn Akkumulator von Null verschieden ist	2	2

Transferbefehle, Sprungbefehle-2:

CJNE A,direct,rel	Sprung, wenn Direkt-Byte und Akkumulator ungleich	3	2
CJNE A,#data,rel	Sprung, wenn Konstante und Akkumulator ungleich	3	2
CJNE Rn,#data,rel	Sprung, wenn Konstante und Register ungleich	3	2
CJNE @Ri,#data,rel	Sprung,wenn Konstante und Indirekte Byte ungleich	3	2
DJNZ Rn,rel	Register -1 und Sprung, wenn nicht Null	2	2
DJNZ direct,rel	Direkt-Byte -1 und Sprung, wenn nicht Null	3	2
JC rel	Sprung bei gesetztem Carry	2	2
JNC rel	Sprung bei gelöschtem Carry	2	2
JB bit,rel	Sprung bei gesetztem Bit	3	2
JNB bit,rel	Sprung bei gelöschtem Bit	3	2
JBC bit,rel	Sprung bei gesetztem Bit und lösche Bit	3	2
NOP	keine Operation	1	1

Logische Befehle:

ANL A,Rn	Akkumulator und Register	1	1
ANL A,direct	Akkumulator und Direkt-Byte	2	1
ANL A,@Ri	Akkumulator und Indirekt-Byte	1	1
ANL A,#data	Akkumulator und Konstante	2	1
ANL direct,A	Direkt-Byte und Akkumulator	2	1
ANL direct,#data	Direkt-Byte und Konstante	3	2
ORL A,Rn	Akkumulator oder Register	1	1
ORL A,direct	Akkumulator oder Direkt-Byte	2	1
ORL A,@Ri	Akkumulator oder Indirekt-Byte	1	1
ORL A,#data	Akkumulator oder Konstante	2	1
ORL direct,A	Direkt-Byte oder Akkumulator	2	1
ORL direct,#data	Direkt-Byte oder Konstante	3	2
XRL A,Rn	Akkumulator exklusiv oder Register	1	1
XRL A,direct	Akkumulator exklusiv oder Direkt-Byte	2	1
XRL A,@Ri	Akkumulator exklusiv oder Indirekt-Byte	1	1
XRL A,#data	Akkumulator exklusiv oder Konstante	2	1
XRL direct,A	Direkt-Byte exklusiv oder Akkumulator	2	1
XRL direct,#data	Direkt-Byte exklusiv oder Konstante 3 2	3	2
CLR A	Lösche Akkumulator	1	1
CPL A	Komplementiere Akkumulator	1	1
RL A	Rotiere Akkumulator nach links	1	1
RLC A	Rotiere Akkumulator nach links mit Carry	1	1
RR A	Rotiere Akkumulator nach rechts	1	1
RRC A	Rotiere Akkumulator nach rechts mit Carry	1	1
SWAP A	Vertausche Nibbles im Akkumulator	1	1

Befehlsliste des Mikrocontrollers C8051F340 (Auszug aus Manual)

9.1. Instruction Set

The instruction set of the CIP-51 System Controller is fully compatible with the standard MCS-51™ instruction
set. Standard 8051 development tools can be used to develop software for the CIP-51. All CIP-51 instructions are the binary and functional equivalent of their MCS-51™ counterparts, including opcodes,
addressing modes and effect on PSW flags. However, instruction timing is different than that of the standard
8051.

9.1.1. Instruction and CPU Timing

In many 8051 implementations, a distinction is made between machine cycles and clock cycles, with machine cycles varying from 2 to 12 clock cycles in length. However, the CIP-51 implementation is based solely on clock cycle timing. All instruction timings are specified in terms of clock cycles.
Due to the pipelined architecture of the CIP-51, most instructions execute in the same number of clock cycles as there are program bytes in the instruction. Conditional branch instructions take one less clock cycle to complete when the branch is not taken as opposed to when the branch is taken.
Table 9.1 is theCIP-51 Instruction Set Summary, which includes the mnemonic, number of bytes, and number of clock cycles for each instruction.

9.1.2. MOVX Instruction and Program Memory

The MOVX instruction is typically used to access external data memory (Note: the C8051F340/1/2/3/4/5/6/ 7 does not support off-chip data or program memory). In the CIP-51, the MOVX write instruction is used to accesses external RAM (XRAM) and the on-chip program memory space implemented as re-programmable Flash memory. The Flash access feature provides a mechanism for the CIP-51 to update program code and use the program memory space for nonvolatile data storage.

Table 9.1. CIP-51 Instruction Set Summary

Mnemonic	Description	Bytes	Clock Cycles
Arithmetic Operations			
ADD A, Rn	Add register to A	1	1
ADD A, direct	Add direct byte to A	2	2
ADD A, @Ri	Add indirect RAM to A	1	2
ADD A, #data	Add immediate to A	2	2
ADDC A, Rn	Add register to A with carry	1	1
ADDC A, direct	Add direct byte to A with carry	2	2
ADDC A, @Ri	Add indirect RAM to A with carry	1	2
ADDC A, #data	Add immediate to A with carry	2	2
SUBB A, Rn	Subtract register from A with borrow	1	1
SUBB A, direct	Subtract direct byte from A with borrow	2	2
SUBB A, @Ri	Subtract indirect RAM from A with borrow	1	2
SUBB A, #data	Subtract immediate from A with borrow	2	2
INC A	Increment A	1	1
INC Rn	Increment register	1	1
INC direct	Increment direct byte	2	2
INC @Ri	Increment indirect RAM	1	2
DEC A	Decrement A	1	1
DEC Rn	Decrement register	1	1
DEC direct	Decrement direct byte	2	2
DEC @Ri	Decrement indirect RAM	1	2
INC DPTR	Increment Data Pointer	1	1
MUL AB	Multiply A and B	1	4
DIV AB	Divide A by B	1	8
DA A	Decimal adjust A	1	1
Logical Operations			
ANL A, Rn	AND Register to A	1	1
ANL A, direct	AND direct byte to A	2	2
ANL A, @Ri	AND indirect RAM to A	1	2
ANL A, #data	AND immediate to A	2	2
ANL direct, A	AND A to direct byte	2	2
ANL direct, #data	AND immediate to direct byte	3	3
ORL A, Rn	OR Register to A	1	1

Table 9.1. CIP-51 Instruction Set Summary (Continued)

Mnemonic	Description	Bytes	Clock Cycles
ORL A, direct	OR direct byte to A	2	2
ORL A, @Ri	OR indirect RAM to A	1	2
ORL A, #data	OR immediate to A	2	2
ORL direct, A	OR A to direct byte	2	2
ORL direct, #data	OR immediate to direct byte	3	3
XRL A, Rn	Exclusive-OR Register to A	1	1
XRL A, direct	Exclusive-OR direct byte to A	2	2
XRL A, @Ri	Exclusive-OR indirect RAM to A	1	2
XRL A, #data	Exclusive-OR immediate to A	2	2
XRL direct, A	Exclusive-OR A to direct byte	2	2
XRL direct, #data	Exclusive-OR immediate to direct byte	3	3
CLR A	Clear A	1	1
CPL A	Complement A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through Carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through Carry	1	1
SWAP A	Swap nibbles of A	1	1
Data Transfer			
MOV A, Rn	Move Register to A	1	1
MOV A, direct	Move direct byte to A	2	2
MOV A, @Ri	Move indirect RAM to A	1	2
MOV A, #data	Move immediate to A	2	2
MOV Rn, A	Move A to Register	1	1
MOV Rn, direct	Move direct byte to Register	2	2
MOV Rn, #data	Move immediate to Register	2	2
MOV direct, A	Move A to direct byte	2	2
MOV direct, Rn	Move Register to direct byte	2	2
MOV direct, direct	Move direct byte to direct byte	3	3
MOV direct, @Ri	Move indirect RAM to direct byte	2	2
MOV direct, #data	Move immediate to direct byte	3	3
MOV @Ri, A	Move A to indirect RAM	1	2
MOV @Ri, direct	Move direct byte to indirect RAM	2	2
MOV @Ri, #data	Move immediate to indirect RAM	2	2
MOV DPTR, #data16	Load DPTR with 16-bit constant	3	3
MOVC A, @A+DPTR	Move code byte relative DPTR to A	1	3
MOVC A, @A+PC	Move code byte relative PC to A	1	3
MOVX A, @Ri	Move external data (8-bit address) to A	1	3
MOVX @Ri, A	Move A to external data (8-bit address)	1	3
MOVX A, @DPTR	Move external data (16-bit address) to A	1	3
MOVX @DPTR, A	Move A to external data (16-bit address)	1	3
PUSH direct	Push direct byte onto stack	2	2
POP direct	Pop direct byte from stack	2	2
XCH A, Rn	Exchange Register with A	1	1
XCH A, direct	Exchange direct byte with A	2	2

Table 9.1. CIP-51 Instruction Set Summary (Continued)

Mnemonic	Description	Bytes	Clock Cycles
XCH A, @Ri	Exchange indirect RAM with A	1	2
XCHD A, @Ri	Exchange low nibble of indirect RAM with A	1	2
Boolean Manipulation			
CLR C	Clear Carry	1	1
CLR bit	Clear direct bit	2	2
SETB C	Set Carry	1	1
SETB bit	Set direct bit	2	2
CPL C	Complement Carry	1	1
CPL bit	Complement direct bit	2	2
ANL C, bit	AND direct bit to Carry	2	2
ANL C, /bit	AND complement of direct bit to Carry	2	2
ORL C, bit	OR direct bit to carry	2	2
ORL C, /bit	OR complement of direct bit to Carry	2	2
MOV C, bit	Move direct bit to Carry	2	2
MOV bit, C	Move Carry to direct bit	2	2
JC rel	Jump if Carry is set	2	2/4
JNC rel	Jump if Carry is not set	2	2/4
JB bit, rel	Jump if direct bit is set	3	3/5
JNB bit, rel	Jump if direct bit is not set	3	3/5
JBC bit, rel	Jump if direct bit is set and clear bit	3	3/5
Program Branching			
ACALL addr11	Absolute subroutine call	2	4
LCALL addr16	Long subroutine call	3	5
RET	Return from subroutine	1	6
RETI	Return from interrupt	1	6
AJMP addr11	Absolute jump	2	4
LJMP addr16	Long jump	3	5
SJMP rel	Short jump (relative address)	2	4
JMP @A+DPTR	Jump indirect relative to DPTR	1	4
JZ rel	Jump if A equals zero	2	2/4
JNZ rel	Jump if A does not equal zero	2	2/4
CJNE A, direct, rel	Compare direct byte to A and jump if not equal	3	3/5
CJNE A, #data, rel	Compare immediate to A and jump if not equal	3	3/5
CJNE Rn, #data, rel	Compare immediate to Register and jump if not equal	3	3/5
CJNE @Ri, #data, rel	Compare immediate to indirect and jump if not equal	3	4/6
DJNZ Rn, rel	Decrement Register and jump if not zero	2	2/4
DJNZ direct, rel	Decrement direct byte and jump if not zero	3	3/5
NOP	No operation	1	1

Notes on Registers, Operands and Addressing Modes:

Rn - Register R0-R7 of the currently selected register bank.

@Ri - Data RAM location addressed indirectly through R0 or R1.

rel - 8-bit, signed (two's complement) offset relative to the first byte of the following instruction. Used by SJMP and all conditional jumps.

direct - 8-bit internal data location's address. This could be a direct-access Data RAM location (0x00-0x7F) or an SFR (0x80-0xFF).

#data - 8-bit constant

#data16 - 16-bit constant

bit - Direct-accessed bit in Data RAM or SFR

addr11 - 11-bit destination address used by ACALL and AJMP. The destination must be within the same 2K-byte page of program memory as the first byte of the following instruction.

addr16 - 16-bit destination address used by LCALL and LJMP. The destination may be anywhere within the 8K-byte program memory space.

There is one unused opcode (0xA5) that performs the same function as NOP.