A

Summary of the 6809 Instruction Set

Appendix A uses the following symbols:

The registers:

A, B	Accumulators
D	Double Accumulator (A and B concatenated, A high-order)
DP	Direct Page Register
X, Y	Index registers
PC	Program Counter
S	Hardware Stack Pointer
U	User Stack Pointer
CC	Status (Condition Code) Register

The flags (statuses), starting with bit 0 of the condition code register and proceeding to bit 7:

C	Carry (Borrow) flag
V	Overflow flag
Z	Zero flag
N	Sign (Negative) flag
I	(Regular) Interrupt Mask bit
H	Half-Carry flag
F	Fast Interrupt Mask bit
E	Entire flag

Symbols in the Status (flags) columns:

(blank)	Operation does not affect flag
X	Operation affects flag
0	Operation clears flag
1	Operation sets flag

Other symbols and abbreviations:

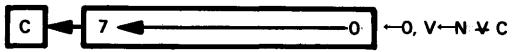
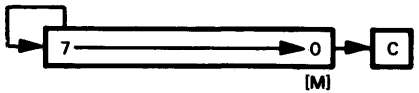
ACx	An accumulator, either Accumulator A or Accumulator B
adr8	An 8-bit address, a 1-byte quantity which may be used to directly address memory locations on the base (direct) page
adr16	A 16-bit memory address
b0-b7	Bits of a Post Byte or an 8-bit register
C	Contents of the Carry flag, either 0 or 1
data8	An 8-bit unit of binary data
data16	A 16-bit unit of binary data
disp8	An 8-bit signed binary address displacement
disp16	A 16-bit signed binary address displacement
EA	Effective address calculated by any addressing method
M	Memory address as determined by base page direct, extended direct, indexed, or indirect addressing
[M]	Contents of M
[M] : [M + 1]	16-bit data item; its high-order byte is the contents of M, and its low-order byte is the contents of the next higher address.
reg	A 16-bit index register or stack pointer (S, U, X, or Y)
reg. list	A list of registers to be stored on or retrieved from a stack
R16	A 16-bit register (D, S, U, X, or Y)
R1, R2	Two registers, both 8-bit or both 16-bit
SP	A stack pointer (either S or U)
ind. forms	Any of the indexed or indirect addressing methods described in Appendix B
[interrupt vector]	The address contained in one of the interrupt vectors (see Table 15-1)
xx(HI)	The high-order 8 bits of the 16-bit quantity xx
xx(LO)	The low-order 8 bits of the 16-bit quantity xx
[ ]	Contents of location enclosed by brackets
[[ ]]	Implied memory address: the contents of the memory location designated by the contents of a register
∧	Logical AND
∨	Logical (Inclusive) OR
⊻	Logical Exclusive-OR
←	Data is transferred in the direction of the arrow
←→	Data is transferred in both directions simultaneously, thus exchanging the contents of the source and the destination.

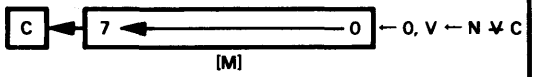
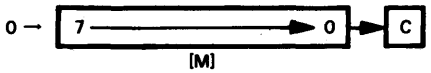
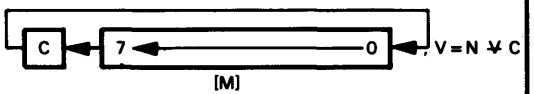
Appendix A. A Summary of the 6809 Instruction Set

Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
Primary Memory Reference and I/O													**The 6809 permits the following addressing modes for all Primary Memory Reference instructions: base page direct, extended direct, indexed, and indirect.**
	LDA LDB	adr8 adr16 ind. forms	2 3 2+	4 5 4+					X	X	0		[ACx] ← [M] Load Accumulator A or B from specified memory location.
	STA STB	adr8 adr16 ind. forms	2 3 2+	4 5 4+					X	X	0		[M] ← [ACx] Store contents of Accumulator A or B in specified memory location.
	LDD	adr8 adr16 ind. forms	2 3 2+	5 6 5+					X	X	0		[D] ← [M]:[M + 1] Load double Accumulator from specified memory location. Sign flag (N) takes the value of bit 15 of the data (bit 7 of Accumulator A).
	STD	adr8 adr16 ind. forms	2 3 2+	5 6 5+					X	X	0		[M]:[M + 1] ← [D] Store contents of double Accumulator in specified memory location. Sign flag takes the value of bit 15 of the data (bit 7 of Accumulator A).
	LDX LDU	adr8 adr16 ind. forms	2 3 2+	5 6 5+					X	X	0		[reg] ← [M]:[M + 1] Load specified register (X, Y, U, or S) from memory. Sign flag (N) takes the value of bit 15 of the data.
	LDY LDS	adr8 adr16 ind. forms	3 4 3+	6 7 6+					X	X	0		
	STX STU	adr8 adr16 ind. forms	2 3 2+	5 6 5+					X	X	0		[M]:[M + 1] ← [reg] Store contents of specified register (X, Y, U, or S) in memory. Sign flag (N) takes the value of bit 15 of the register.
	STY STS	adr8 adr16 ind. forms	3 4 3+	6 7 6+					X	X	0		

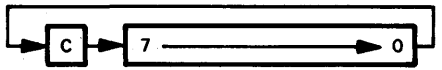
Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
Secondary Memory Reference (Memory Operate)													**The 6809 permits the following addressing modes for all Secondary Memory Reference instructions: base page direct, extended direct, indexed, and indirect.**
	ADCA ADCB	adr8 adr16 ind. forms	2 3 2+	4 5 4+			X		X	X	X	X	[ACx] ← [ACx] + [M] + C Add with carry to Accumulator A or B.
	ADDA ADDB	adr8 adr16 ind. forms	2 3 2+	4 5 4+			X		X	X	X	X	[ACx] ← [ACx] + [M] Add contents of specified memory location to Accumulator A or B.
	ADDD	adr8 adr16 ind. forms	2 3 2+	6 7 6+					X	X	X	X	[D] ← [D] + [M]:[M + 1] Add 16-bit value from memory to double Accumulator. The operand's high-order byte is in the specified memory location; the low-order byte is in the next higher address.
	ANDA ANDB	adr8 adr16 ind. forms	2 3 2+	4 5 4+					X	X	0		[ACx] ← [ACx] Λ [M] AND contents of specified memory location with Accumulator A or B.
	BITA BITB	adr8 adr16 ind. forms	2 3 2+	4 5 4+					X	X	0		[ACx] Λ [M] AND contents of specified memory location with Accumulator A or B. Only the Status register is affected.
	CMPA CMPB	adr8 adr16 ind. forms	2 3 2+	4 5 4+			X		X	X	X	X	[ACx] − [M] Compare contents of specified memory location with Accumulator A or B. Only the Status register is affected.
	CMPD	adr8 adr16 ind. forms	3 4 3+	7 8 7+					X	X	X	X	[D] − [M]:[M + 1] Compare 16-bit data with double Accumulator. Only the status register is affected. The high-order byte of the data is in the specified memory location; the low-order byte is in the next higher address.
	CMPS CMPU CMPY	adr8 adr16 ind. forms	3 4 3+	7 8 7+					X	X	X	X	[reg] − [M]:[M + 1] Compare 16-bit data with specified register (S, U, X, or Y). Only the status register is affected. The high-order byte of the data is in the specified memory location; the low-order byte is in the next higher address.

Appendix A. A Summary of the 6809 Instruction Set (Continued)

Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
Secondary Memory Reference (Memory Operate) (Continued)	CMPX	adr8 adr16 ind. forms	2 3 2+	6 7 6+					X	X	X	X	Same as CMPS/CMPU/CMPY. See page A-4.
	EORA EORB	adr8 adr16 ind. forms	2 3 2+	4 5 4+					X	X	0		[ACx] ← [ACx] ⊻ [M] Logical Exclusive-OR contents of specified memory location with Accumulator A or B.
	ORA ORB	adr8 adr16 ind. forms	2 3 2+	4 5 4+					X	X	0		[ACx] ← [ACx] V [M] Logical (Inclusive) OR contents of specified memory location with Accumulator A or B.
	SBCA SBCB	adr8 adr16 ind. forms	2 3 2+	4 5 4+			X		X	X	X	X	[ACx] ← [ACx] − [M] − C Subtract contents of specified memory location and contents of Carry flag from Accumulator A or B.
	SUBA SUBB	adr8 adr16 ind. forms	2 3 2+	4 5 4+			X		X	X	X	X	[ACx] ← [ACx] − [M] Subtract contents of specified memory location from Accumulator A or B.
	SUBD	adr8 adr16 ind. forms	2 3 2+	6 7 6+					X	X	X	X	[D] ← [D] − [M]:[M + 1] Subtract 16-bit value in memory from double Accumulator. The operand's high-order byte is in the specified memory address; the low-order byte is in the next higher address.
	ASL	adr8 adr16 ind. forms	2 3 2+	6 7 6+			X		X	X	X	X	 Arithmetic shift left. Bit 0 is set to 0.
	ASR	adr8 adr16 ind. forms	2 3 2+	6 7 6+			X		X	X		X	 Arithmetic shift right. Bit 7 stays the same.

Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
Secondary Memory Reference (Memory Operate) (Continued)	CLR	adr8 adr16 ind. forms	2 3 2+	6 7 6+					0	1	0	0	$[M] \leftarrow 00_{16}$ Clear specified memory location.
	COM	adr8 adr16 ind. forms	2 3 2+	6 7 6+					X	X	0	1	$[M] \leftarrow \overline{[M]}$ Ones complement contents of memory location.
	DEC	adr8 adr16 ind. forms	2 3 2+	6 7 6+					X	X	X		[M] ← [M]−1 Decrement (by 1) contents of memory location.
	INC	adr8 adr16 ind. forms	2 3 2+	6 7 6+					X	X	X		[M] ← [M]+1 Increment (by 1) contents of memory location.
	LSL	adr8 adr16 ind. forms	2 3 2+	6 7 6+			X		X	X	X	X	 Logical shift left. Same as ASL.
	LSR	adr8 adr16 ind. forms	2 3 2+	6 7 6+					0	X		X	 Logical shift right. Bit 7 is set to 0.
	NEG	adr8 adr16 ind. forms	2 3 2+	6 7 6+			X		X	X	X	X	$[M] \leftarrow 00_{16} - [M]$ Twos complement (negate) contents of memory location. Set Carry if result is 00_{16} and clear Carry otherwise. Set Overflow if result is 80_{16} and clear Overflow otherwise.
	ROL	adr8 adr16 ind. forms	2 3 2+	6 7 6+					X	X	X	X	 Rotate contents of memory location left through Carry flag.

Appendix A. A Summary of the 6809 Instruction Set (Continued)

Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
Secondary Memory Reference (Memory Operate) (Continued)	ROR	adr8 adr16 ind. forms	2 3 2+	6 7 6+					X	X		X	 [M] Rotate contents of memory location right through Carry flag.
	TST	adr8 adr16 ind. forms	2 3 2+	6 7 6+					X	X	0		[M] – 00_{16} Test contents of memory location for zero or negative value.
Immediate	LDA LDB	data8	2	2					X	X	0		[ACx] ← data8 Load Accumulator A or B immediate.
	LDD	data16	3	3					X	X	0		[D] ← data16 Load double Accumulator immediate. Sign flag reflects bit 15 of the data (bit 7 of Accumulator A).
	LDU LDX	data16	3	3					X	X	0		[reg] ← data16 Load specified register (X,Y,U, or S) immediate. Sign flag (N) reflects bit 15 of the register.
	LDS LDY	data16	4	4					X	X	0		
Immediate Operate	ADCA ADCB	data8	2	2			X		X	X	X	X	[ACx] ← [ACx] + data8 + C Add immediate with carry to Accumulator A or B.
	ADDA ADDB	data8	2	2			X		X	X	X	X	[ACx] ← [ACx] + data8 Add immediate to Accumulator A or B.
	ADDD	data16	3	4					X	X	X	X	[D] ← [D] + data16 Add 16-bit data to double Accumulator. The high-order byte follows the operation code; the low-order byte follows the high-order byte.

Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
(Immediate Operate (Continued)	ANDA ANDB	data8	2	2					X	X	0		[ACx] ← [ACx] Λ data8 Logical AND immediate with Accumulator A or B.
	BITA BITB	data8	2	2					X	X	0		[ACx] Λ data8 Logical AND immediate with Accumulator A or B but affect only the status register.
	CMPA CMPB	data8	2	2			X		X	X	X	X	[ACx] – data8 Subtract immediate from Accumulator A or B but affect only the status register.
	CMPD	data16	4	5					X	X	X	X	[D] – data16 Subtract immediate from double Accumulator but affect only the status register.
	CMPS CMPU CMPY	data16	4	5					X	X	X	X	[reg] – data16 Subtract immediate from specified register (S, U, X, or Y), but affect only the status register.
	CMPX	data16	3	4					X	X	X	X	
	EORA EORB	data8	2	2					X	X	0		[ACx] ← [ACx] ⊻ data8 Logical Exclusive-OR immediate with Accumulator A or B.
	ORA ORB	data8	2	2					X	X	0		[ACx] ← [ACx] V data8 Logical (Inclusive) OR immediate with Accumulator A or B.
	SBCA SBCB	data8	2	2			X		X	X	X	X	[ACx] ← [ACx] – data8 – C Subtract with borrow (carry) immediate from Accumulator A or B.
	SUBA SUBB	data8	2	2			X		X	X	X	X	[ACx] ← [ACx] – data8 Subtract immediate from Accumulator A or B.
	SUBD	data16	3	4					X	X	X	X	[D] ← [D] – data16 Subtract immediate from double Accumulator D.

Appendix A. A Summary of the 6809 Instruction Set (Continued)

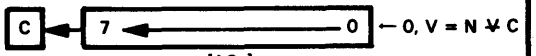
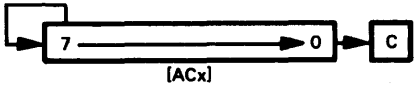
Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
Jump	BRA	disp8	2	3									[PC] ← [PC] + disp8 + 2 Unconditional branch relative to current contents of Program Counter.
	JMP	adr8 adr16 ind. forms	2 3 2+	3 4 3+									[PC] ← EA Unconditional jump to the specified (effective) address using base page direct, extended direct, indexed, or indirect addressing.
	LBRA	disp16	3	5									[PC] ← [PC] + disp16 + 3 Unconditional long branch relative to current contents of Program Counter.
	TFR	R16,PC	2	7									[PC] ← [R16] Unconditional jump to the address in the specified 16-bit register (D, S, U, X, or Y).
Subroutine Call and Return	BSR	disp8	2	7									[[S]−1] ← [PC(LO)] [[S]−2] ← [PC(HI)] [S] ← [S] − 2 [PC] ← [PC] + disp8 + 2 Unconditional branch to subroutine relative to current contents of Program Counter, saving current Program Counter in the Hardware Stack before performing branch.
	EXG	R16,PC	2	8									[R16] ⟷ [PC] Unconditional jump to the address in the specified 16-bit register (D, S, U, X, or Y), save current Program Counter in the specified register. Can be used to call a subroutine or return from a subroutine; the specified 16-bit register acts as a link.
	JSR	adr8 adr16 ind. forms	2 3 2+	7 8 7+									[[S] − 1] ← [PC(LO)] [[S] − 2] ← [PC(HI)] [S] ← [S] − 2 [PC] ← EA Unconditional jump to subroutine at the specified (effective) address using base page direct, extended direct, indirect, or indexed addressing. Saves current Program Counter in the Hardware Stack before performing jump.

Appendix A. A Summary of the 6809 Instruction Set (Continued)

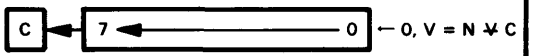
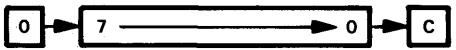
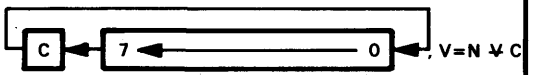
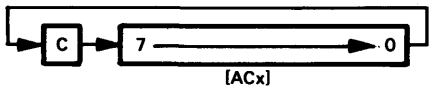
Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
(Subroutine Call and Return (Continued)	LBSR	disp16	3	9									[[S] – 1] ← [PC(LO)] [[S] – 2] ← [PC(HI)] [S] ← [S] – 2 [PC] ← [PC] + disp16 + 3 Unconditional long branch to subroutine relative to current contents of Program Counter, saving current Program Counter in the Hardware Stack before performing branch.
	PULS PULU	PC, reg. list	2	5+									Return from Subroutine and load other registers from Hardware or User Stack as specified in post byte. Bit 7 of the post byte must be 1 so that the Program Counter is among the registers loaded from the Stack. See the Stack functions section of this table for a description of PULS and PULU operation.
	RTS		1	5									[PC(HI)] ← [[S]] [PC(LO)] ← [[S] + 1] [S] ← [S] + 2 Return from subroutine: remove program counter from top of Hardware Stack and increment Hardware Stack Pointer twice.
Branch on Condition													[PC] ← [PC] + disp8 + 2 if the given condition is true:
	BCC	disp8	2	3									C = 0
	BCS	disp8	2	3									C = 1
	BEQ	disp8	2	3									Z = 1
	BGE	disp8	2	3									N ⊻ V = 0
	BGT	disp8	2	3									Z V(N ⊻ V) = 0
	BHI	disp8	2	3									C V Z = 0
	BHS	disp8	2	3									C = 0
	BLE	disp8	2	3									Z V(N ⊻ V) = 1
	BLO	disp8	2	3									C = 1
	BLS	disp8	2	3									C V Z = 1
	BLT	disp8	2	3									N ⊻ V = 1
	BMI	disp8	2	3									N = 1
	BNE	disp8	2	3									Z = 0
	BPL	disp8	2	3									N = 0

Appendix A. A Summary of the 6809 Instruction Set (Continued)

Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
Branch on Condition (Continued)	BVC	disp8	2	3									V = 0
	BVS	disp8	2	3									V = 1
													[PC] ← [PC] + 2 if the given condition is not true. Note that BHS and BCC are different mnemonics for the same operation code, as are BLO and BCS.
													[PC] ← [PC] + disp16 + 4 if the given condition is true:
	LBCC	disp16	4	5(6)									C = 0
	LBCS	disp16	4	5(6)									C = 1
	LBEQ	disp16	4	5(6)									Z = 1
	LBGE	disp16	4	5(6)									N ⊻ V = 0
	LBGT	disp16	4	5(6)									Z V(N ⊻ V) = 0
	LBHI	disp16	4	5(6)									C V Z = 0
	LBHS	disp16	4	5(6)									C = 0
	LBLE	disp16	4	5(6)									Z V(N ⊻ V) = 1
	LBLO	disp16	4	5(6)									C = 1
	LBLS	disp16	4	5(6)									C V Z = 1
	LBLT	disp16	4	5(6)									N ⊻ V = 1
	LBMI	disp16	4	5(6)									N = 1
	LBNE	disp16	4	5(6)									Z = 0
	LBPL	disp16	4	5(6)									N = 0
	LBVC	disp16	4	5(6)									V = 0
	LBVS	disp16	4	5(6)									V = 1
													[PC] ← [PC] + 4 if the given condition is not true. Note that LBHS and LBCC are different mnemonics for the same operation code, as are LBLO and LBCS. A long branch instruction takes 6 cycles to execute if it performs the branch and 5 cycles otherwise.
Register to Register Move	EXG	R1, R2	2	8									[R1] ⟷ [R2] Exchange contents of specified registers. No effect on Status register (CC) unless R1 or R2 is Status register.
	TFR	R1, R2	2	7									[R2] ← [R1] Transfer contents of R1 to R2. No effect on Status register (CC) unless R2 is CC.

Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
Register-Register Operate	ABX		1	3									[X] ← [X] + [B] Add unsigned contents of Accumulator B to Index Register X.
	MUL		1	11						X		X	[D] ← [A] × [B] Multiply unsigned numbers in Accumulators A and B and place result in D. Carry flag takes the value of bit 7 of Accumulator B.
	SEX		1	2					X	X			[A] ← FF_{16} if bit 7 of Accumulator B is 1. [A] ← 00_{16} if bit 7 of Accumulator B is 0. Transform an 8-bit twos complement number in B into a 16-bit twos complement number in D.
Register Operate	ASLA ASLB		1	2			X		X	X	X	X	 Arithmetic shift left Accumulator A or B. Bit 0 is set to 0.
	ASRA ASRB		1	2			X		X	X		X	 Arithmetic shift right Accumulator A or B. Bit 7 stays the same.
	CLRA CLRB		1	2					0	1	0	0	[ACx] ← 00_{16} Clear Accumulator A or B.
	COMA COMB		1	2					X	X	0	1	[ACx] ← $\overline{[ACx]}$ Ones complement contents of Accumulator A or B.
	DAA		1	2					X	X	X	X	Decimal adjust Accumulator A. Convert contents of Accumulator A (assumed to be the binary sum of BCD operands) to BCD format. Carry is set if it was previously set or if the adjustment results in a carry.
	DECA DECB		1	2					X	X	X		[ACx] ← [ACx] − 1 Decrement (by 1) contents of Accumulator A or B. Set Overflow flag if result is $7F_{16}$ and clear Overflow flag otherwise.

Appendix A. A Summary of the 6809 Instruction Set (Continued)

Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
Register Operate (Continued)	INCA INCB		1	2					X	X	X		[ACx] ← [ACx] + 1 Increment (by 1) contents of Accumulator A or B. Set Overflow flag if result is 80_{16} and clear Overflow flag otherwise.
	LSLA LSLB		1	2			X		X	X	X	X	 Logical shift left Accumulator A or B. Bit 0 is set to 0. Same as ASL.
	LSRA LSRB		1	2					0	X		X	 Logical shift right Accumulator A or B. Bit 7 is set to 0.
	NEGA NEGB		1	2			X		X	X	X	X	[ACx] ← 00_{16} – [ACx] Twos complement (negate) contents of Accumulator A or B. Set Carry flag if result is 00_{16} and clear Carry flag otherwise. Set Overflow flag if result is 80_{16} and clear Overflow flag otherwise.
	ROLA ROLB		1	2					X	X	X	X	 Rotate Accumulator A or B left through Carry flag.
	RORA RORB		1	2					X	X		X	 Rotate Accumulator A or B right through Carry flag.

Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
Register Operate (Continued)	TSTA TSTB		1	2					X	X	0		[ACx] – 00_{16} Test contents of Accumulator A or B for zero or negative value.
	LEAX LEAY	ind. forms	2+	4+						X			[reg] ← EA Form the effective address according to any of the indexed/indirect addressing modes (see Appendix B) and load that address into the specified register (X, Y, S, or U). LEA instructions are primarily intended to calculate an effective address once for repeated use, but may also be employed to perform 16-bit arithmetic.
	LEAS LEAU	ind. forms	2+	4+									
Stack	PSHS PSHU	reg. list	2	5+									Test post byte and store registers in specified stack as follows: Condition: b7 = 1; [SP] ← [SP] – 1, [[SP]] ← [PC(LO)] [SP] ← [SP] – 1, [[SP]] ← [PC(HI)] b6 = 1; [SP] ← [SP] – 1, [[SP]] ← [U(LO)] or [S(LO)] [SP] ← [SP] – 1, [[SP]] ← [U(HI)] or [S(HI)] b5 = 1; [SP] ← [SP] – 1, [[SP]] ← [Y(LO)] [SP] ← [SP] – 1, [[SP]] ← [Y(HI)] b4 = 1; [SP] ← [SP] – 1, [[SP]] ← [X(LO)] [SP] ← [SP] – 1, [[SP]] ← [X(HI)] b3 = 1; [SP] ← [SP] – 1, [[SP]] ← [DP] b2 = 1; [SP] ← [SP] – 1, [[SP]] ← [B] b1 = 1; [SP] ← [SP] – 1, [[SP]] ← [A] b0 = 1; [SP] ← [SP] – 1, [[SP]] ← [CC] Push all, none, or any subset of registers onto the specified Stack, except for the pointer to that Stack. Execution time increases by one cycle for each byte pushed.

Appendix A. A Summary of the 6809 Instruction Set (Continued)

Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
Stack (Continued)	PULS PULU	reg. list	2	5+									Test post byte and load registers from specified stack as follows: Condition: b0 = 1; [CC] ← [[SP]], [SP] ← [SP] + 1 b1 = 1; [A] ← [[SP]], [SP] ← [SP] + 1 b2 = 1; [B] ← [[SP]], [SP] ← [SP] + 1 b3 = 1; [DP] ← [[SP]], [SP] ← [SP] + 1 b4 = 1; [X(HI)] ← [[SP]], [SP] ← [SP] + 1 [X(LO)] ← [[SP]], [SP] ← [SP] + 1 b5 = 1; [Y(HI)] ← [[SP]], [SP] ← [SP] + 1 [Y(LO)] ← [[SP]], [SP] ← [SP] + 1 b6 = 1; [U(HI)] or [S(HI)] ← [[SP]], [SP] ← [SP] + 1 [U(LO)] or [S(LO)] ← [[SP]], [SP] ← [SP] + 1 b7 = 1; [PC(HI)] ← [[SP]], [SP] ← [SP] + 1 [PC(LO)] ← [[SP]], [SP] ← [SP] + 1 Pull all, none, or any subset of registers from the specified stack, except for the Pointer to that Stack. Status register bits are determined by byte pulled from Stack. Execution time increases by one cycle for each byte pulled.

Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
Interrupt	CWAI	data8	2	20									[CC] ← [CC] Λ data8. This may clear CC bits. E ← 1 [S] ← [S] – 1, [[S]] ← [PC(LO)] [S] ← [S] – 1, [[S]] ← [PC(HI)] [S] ← [S] – 1, [[S]] ← [U(LO)] [S] ← [S] – 1, [[S]] ← [U(HI)] [S] ← [S] – 1, [[S]] ← [Y(LO)] [S] ← [S] – 1, [[S]] ← [Y(HI)] [S] ← [S] – 1, [[S]] ← [X(LO)] [S] ← [S] – 1, [[S]] ← [X(HI)] [S] ← [S] – 1, [[S]] ← [DP] [S] ← [S] – 1, [[S]] ← [B] [S] ← [S] – 1, [[S]] ← [A] [S] ← [S] – 1, [[S]] ← [CC] Stores all registers in Hardware Stack and waits for an interrupt. When non-masked interrupt occurs, vectors to corresponding interrupt service routine. Note that a fast interrupt (FIRQ) service routine will be entered with all registers saved, but RTI will restore them correctly since CWAI sets E flag. CWAI does not float the system busses.

Appendix A. A Summary of the 6809 Instruction Set (Continued)

Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
Interrupt (Continued)	RTI		1	6/15									Pull registers from Hardware Stack in accordance with value of E flag in Status register. If E = 0, pull the subset: [CC] ← [[S]], [S] ← [S] + 1 [PC(HI)] ← [[S]], [S] ← [S] + 1 [PC(LO)] ← [[S]], [S] ← [S] + 1 If E = 1, pull the full complement of registers: [CC] ← [[S]], [S] ← [S] + 1 [A] ← [[S]], [S] ← [S] + 1 [B] ← [[S]], [S] ← [S] + 1 [DP] ← [[S]], [S] ← [S] + 1 [X(HI)] ← [[S]], [S] ← [S] + 1 [X(LO)] ← [[S]], [S] ← [S] + 1 [Y(HI)] ← [[S]], [S] ← [S] + 1 [Y(LO)] ← [[S]], [S] ← [S] + 1 [U(HI)] ← [[S]], [S] ← [S] + 1 [U(LO)] ← [[S]], [S] ← [S] + 1 [PC(HI)] ← [[S]], [S] ← [S] + 1 [PC(LO)] ← [[S]], [S] ← [S] + 1 Status register bits are as removed from the Hardware Stack.
	SWI		1	19	1	1							Save all registers in the Hardware Stack and transfer control to interrupt subroutine. Vectors are in: FFFA and FFFB for SWI FFF4 and FFF5 for SWI2 FFF2 and FFF3 for SWI3 E ← 1 [S] ← [S] − 1, [[S]] ← [PC(LO)] [S] ← [S] − 1, [[S]] ← [PC(HI)] [S] ← [S] − 1, [[S]] ← [U(LO)] [S] ← [S] − 1, [[S]] ← [U(HI)] [S] ← [S] − 1, [[S]] ← [Y(LO)] [S] ← [S] − 1, [[S]] ← [Y(HI)] [S] ← [S] − 1, [[S]] ← [X(LO)] [S] ← [S] − 1, [[S]] ← [X(HI)]
	SWI2		2	20	1								
	SWI3		2	20	1								

Type	Mnemonic	Operand(s)	Bytes	Cycles	Status								Operation Performed
					E	F	H	I	N	Z	V	C	
Interrupt (Continued)	SYNC		1	2									[S] ← [S] − 1, [[S]] ← [DP] [S] ← [S] − 1, [[S]] ← [B] [S] ← [S] − 1, [[S]] ← [A] [S] ← [S] − 1, [[S]] ← [CC] [PC] ← [interrupt vector] Note that the SWI disables the regular and fast interrupts, whereas SWI2 and SWI3 do not affect either one. Stop processing instructions. Float system busses and wait for an interrupt. When an interrupt occurs, resume processing as follows: a. If interrupt is enabled, transfer control to the service routine. b. If interrupt is disabled, continue execution at next instruction in sequence.
Status	ANDCC	data8	2	3									[CC] ← [CC] Λ data8 Logically AND immediate data with contents of status register. Used to clear bits of status register by logically ANDing them with '0's.
	ORCC	data8	2	3									[CC] ← [CC] V data8 Logically (Inclusive) OR immediate data with contents of status register. Used to set bits of status register by logically ORing them with '1's.
No Operations	BRN	disp8	2	3									Branch never. This is a No operation.
	LBRN	disp16	5	4									Long branch never. This is a No operation.
	NOP		2	1									No operation.

B

Summary of 6809 Indexed and Indirect Addressing Modes

Type	Form	Non-indirect		Cycles	Bytes	Indirect		Cycles	Bytes
		Assembler Form	Post-Byte Op-code	+	+	Assembler Form	Post-Byte Op-code	+	+
Constant Offset from R	No Offset	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5-Bit Offset	n,R	0RRnnnnn	1	0		Defaults to 8-bit		
	8-Bit Offset	nn,R	1RR01000	1	1	[nn,R]	1RR11000	4	1
	16-Bit Offset	mmnn,R	1RR01001	4	2	[mmnn,R]	1RR11001	7	2
Accumulator Offset from R	A — Register Offset	A,R	1RR00110	1	0	[A,R]	1RR10110	4	0
	B — Register Offset	B,R	1RR00101	1	0	[B,R]	1RR10101	4	0
	D — Register Offset	D,R	1RR01011	4	0	[D,R]	1RR11011	7	0
Auto Increment/ Decrement R	Increment by 1	,R+	1RR00000	2	0		Not allowed		
	Increment by 2	,R++	1RR00001	3	0	[,R++]	1RR10001	6	0
	Decrement by 1	,-R	1RR00010	2	0		Not allowed		
	Decrement by 2	,--R	1RR00011	3	0	[,--R]	1RR10011	6	0
Constant Offset from PC	8-Bit Offset	label,PCR	1XX01100	1	1	[label,PCR]	1XX11100	4	1
	16-Bit Offset	label,PCR	1XX01101	5	2	[label,PCR]	1XX11101	8	2
Extended Indirect	16-Bit Address	—	—	—	—	[mmnn]	10011111	5	2

R = X, Y, U, or S RR: 00 = X 10 = U
XX = Don't Care 01 = Y 11 = S

Note: This chart conforms to Motorola nomenclature; their use of square brackets [ ] indicates to the assembler that the addressing mode is indirect — thus, their use of [ ] differs from the use in Appendix A.

C

6809 Instruction Codes, Memory Requirements, and Execution Times

Address Mode →	Inherent			Immediate			Direct			Extended			Indexed/Indirect			Relative		
Operand Form →				data8 or data16			adr8			adr16			See Appendix B			label or displacement		
Instruction Mnemonic ↓	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes
ABX	3A	3	1															
ADCA				89	2	2	99	4	2	B9	5	3	A9	4+	2+			
ADCB				C9	2	2	D9	4	2	F9	5	3	E9	4+	2+			
ADDA				8B	2	2	9B	4	2	BB	5	3	AB	4+	2+			
ADDB				CB	2	2	DB	4	2	FB	5	3	EB	4+	2+			
ADDD				C3	4	3	D3	6	2	F3	7	3	E3	6+	2+			
ANDA				84	2	2	94	4	2	B4	5	3	A4	4+	2+			
ANDB				C4	2	2	D4	4	2	F4	5	3	E4	4+	2+			
ANDCC				1C	3	2												
ASL							08	6	2	78	7	3	68	6+	2+			
ASLA	48	2	1															
ASLB	58	2	1															
ASR							07	6	2	77	7	3	67	6+	2+			
ASRA	47	2	1															
ASRB	57	2	1															
BCC																24	3	2
BCS																25	3	2
BEQ																27	3	2
BGE																2C	3	2
BGT																2E	3	2
BHI																22	3	2
BHS																24	3	2
BITA				85	2	2	95	4	2	B5	5	3	A5	4+	2+			
BITB				C5	2	2	D5	4	2	F5	5	3	E5	4+	2+			
BLE																2F	3	2
BLO																25	3	2
BLS																23	3	2
BLT																2D	3	2
BMI																2B	3	2
BNE																26	3	2
BPL																2A	3	2
BRA																20	3	2
BRN																21	3	2
BSR																8D	7	2
BVC																28	3	2
BVS																29	3	2
CLR							0F	6	2	7F	7	3	6F	6+	2+			
CLRA	4F	2	1															
CLRB	5F	2	1															
CMPA				81	2	2	91	4	2	B1	5	3	A1	4+	2+			
CMPB				C1	2	2	D1	4	2	F1	5	3	E1	4+	2+			
CMPD				10 83	5	4	10 93	7	3	10 B3	8	4	10 A3	7+	3+			

Address Mode →	Inherent			Immediate			Direct			Extended			Indexed/Indirect			Relative			notes
Operand Form →				data8 or data16			adr8			adr16			See Appendix B			label or displacement			
Instruction Mnemonic ↓	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	
CMPS				11 8C	5	4	11 9C	7	3	11 BC	8	4	11 AC	7+	3+				
CMPU				11 83	5	4	11 93	7	3	11 B3	8	4	11 A3	7+	3+				
CMPX				8C	4	3	9C	6	2	BC	7	3	AC	6+	2+				
CMPY				10 8C	5	4	10 9C	7	3	10 BC	8	4	10 AC	7+	3+				
COM							03	6	2	73	7	3	63	6+	2+				
COMA	43	2	1																
COMB	53	2	1																
CWAI				3C	20	2													
DAA	19	2	1																
DEC							0A	6	2	7A	7	3	6A	6+	2+				
DECA	4A	2	1																
DECB	5A	2	1																
EORA				88	2	2	98	4	2	B8	5	3	A8	4+	2+				
EORB				C8	2	2	D8	4	2	F8	5	3	E8	4+	2+				
EXG				1E	8	2													
INC							0C	6	2	7C	7	3	6C	6+	2+				2
INCA	4C	2	1																
INCB	5C	2	1																
JMP							0E	3	2	7E	4	3	6E	3+	2+				
JSR							9D	7	2	BD	8	3	AD	7+	2+				
LBCC																10 24	5(6)	4	1
LBCS																10 25	5(6)	4	1
LBEQ																10 27	5(6)	4	1
LBGE																10 2C	5(6)	4	1
LBGT																10 2E	5(6)	4	1
LBHI																10 22	5(6)	4	1
LBHS																10 24	5(6)	4	1
LBLE																10 2F	5(6)	4	1
LBLO																10 25	5(6)	4	1
LBLS																10 23	5(6)	4	1
LBLT																10 2D	5(6)	4	1
LBMI																10 2B	5(6)	4	1
LBNE																10 26	5(6)	4	1
LBPL																10 2A	5(6)	4	1
LBRA																16	5	3	
LBRN																10 21	5	4	
LBSR																17	9	3	
LBVC																10 28	5(6)	4	1
LBVS																10 29	5(6)	4	1
LDA				86	2	2	96	4	2	B6	5	3	A6	4+	2+				
LDB				C6	2	2	D6	4	2	F6	5	3	E6	4+	2+				
LDD				CC	3	3	DC	5	2	FC	6	3	EC	5+	2+				

Address Mode →	Inherent			Immediate			Direct			Extended			Indexed/Indirect			Relative			notes
Operand Form →				data8 or data16			adr8			adr16			See Appendix B			label or displacement			
Instruction Mnemonic ↓	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	
LDS				10 CE	4	4	10 DE	6	3	10 FE	7	4	10 EE	6+	3+				
LDU				CE	3	3	DE	5	2	FE	6	3	EE	5+	2+				
LDX				8E	3	3	9E	5	2	BE	6	3	AE	5+	2+				
LDY				10 8E	4	4	10 9E	6	3	10 BE	7	4	10 AE	6+	3+				
LEAS													32	4+	2+				
LEAU													33	4+	2+				
LEAX													30	4+	2+				
LEAY													31	4+	2+				
LSL							08	6	2	78	7	3	68	6+	2+				
LSLA	48	2	1																
LSLB	58	2	1																
LSR							04	6	2	74	7	3	64	6+	2+				
LSRA	44	2	1																
LSRB	54	2	1																
MUL	3D	11	1																
NEG							00	6	2	70	7	3	60	6+	2+				
NEGA	40	2	1																
NEGB	50	2	1																
NOP	12	2	1																
ORA				8A	2	2	9A	4	2	BA	5	3	AA	4+	2+				
ORB				CA	2	2	DA	4	2	FA	5	3	EA	4+	2+				
ORCC				1A	3	2													
PSHS				34	5+	2													2, 3
PSHU				36	5+	2													2, 3
PULS				35	5+	2													2, 3
PULU				37	5+	2													2, 3
ROL							09	6	2	79	7	3	69	6+	2+				
ROLA	49	2	1																
ROLB	59	2	1																
ROR							06	6	2	76	7	3	66	6+	2+				
RORA	46	2	1																
RORB	56	2	1																
RTI	3B	6/15	1																
RTS	39	5	1																
SBCA				82	2	2	92	4	2	B2	5	3	A2	4+	2+				
SBCB				C2	2	2	D2	4	2	F2	5	3	E2	4+	2+				
SEX	1D	2	1																
STA							97	4	2	B7	5	3	A7	4+	2+				
STB							D7	4	2	F7	5	3	E7	4+	2+				
STD							DD	5	2	FD	6	3	ED	5+	2+				
STS							10 DF	6	3	10 FF	7	4	10 EF	6+	3+				

Address Mode →	Inherent			Immediate			Direct			Extended			Indexed/Indirect			Relative			notes
Operand Form →				data8 or data16			adr8			adr16			See Appendix B			label or displacement			
Instruction Mnemonic ↓	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	
STU							DF	5	2	FF	6	3	EF	5+	2+				
STX							9F	5	2	BF	6	3	AF	5+	2+				
STY							10 9F	6	3	10 BF	7	4	10 AF	6+	3+				
SUBA				80	2	2	90	4	2	B0	5	3	A0	4+	2+				
SUBB				C0	2	2	D0	4	2	F0	5	3	E0	4+	2+				
SUBD				83	4	3	93	6	2	B3	7	3	A3	6+	2+				
SWI	3F	19	1																
SWI2	10 3F	20	2																
SWI3	11 3F	20	2																
SYNC	13	2	1																
TFR				1F	7	2													2
TST							0D	6	2	7D	7	3	6D	6+	2+				
TSTA	4D	2	1																
TSTB	5D	2	1																

Note 1: The cycle count in parentheses applies if the branch is taken.

Note 2: The immediate data in this instruction's object code is always an encoded register specification. See the description of this instruction in Chapter 22 for details.

Note 3: A PSH or PUL instruction requires 5 cycles plus one cycle for each byte pushed or pulled.

D

6809 Instruction Object Codes in Numerical Order

The following symbols and abbreviations appear in this appendix:

adr8	8-bit address
adr16	16-bit address
data8	8-bit data
data16	16-bit data
dd	8-bit data
dd dd	16-bit data
label	The destination of a Jump or Branch
mm	8-bit displacement in the object code
mm nn	16-bit displacement in the object code
pp	post byte for indexed and indirect addressing
qq	8-bit address
ssqq	16-bit address

6809 Instruction Object Codes in Numerical Order

Object Code[1]	Instruction[2, 3]	Addressing Mode
00 qq	NEG adr8	Base page (direct)
03 qq	COM adr8	Base page (direct)
04 qq	LSR adr8	Base page (direct)
06 qq	ROR adr8	Base page (direct)
07 qq	ASR adr8	Base page (direct)
08 qq	ASL adr8 / LSL adr8	Base page (direct)
09 qq	ROL adr8	Base page (direct)
0A qq	DEC adr8	Base page (direct)
0C qq	INC adr8	Base page (direct)
0D qq	TST adr8	Base page (direct)
0E qq	JMP adr8	Base page (direct)
0F qq	CLR adr8	Base page (direct)
10 21 mm nn	LBRN label	Relative
10 22 mm nn	LBHI label	Relative
10 23 mm nn	LBLS label	Relative
10 24 mm nn	LBHS label / LBCC label	Relative
10 25 mm nn	LBLO label / LBCS label	Relative
10 26 mm nn	LBNE label	Relative
10 27 mm nn	LBEQ label	Relative
10 28 mm nn	LBVC label	Relative
10 29 mm nn	LBVS label	Relative
10 2A mm nn	LBPL label	Relative
10 2B mm nn	LBMI label	Relative
10 2C mm nn	LBGE label	Relative
10 2D mm nn	LBLT label	Relative
10 2E mm nn	LBGT label	Relative
10 2F mm nn	LBLE label	Relative
10 3F	SWI2	Inherent
10 83 dd dd	CMPD data16	Immediate
10 8C dd dd	CMPY data16	Immediate
10 8E dd dd	LDY data16	Immediate
10 93 qq	CMPD adr8	Base page (direct)
10 9C qq	CMPY adr8	Base page (direct)
10 9E qq	LDY adr8	Base page (direct)
10 9F qq	STY adr8	Base page (direct)
10 A3 pp[1]	CMPD indexed forms	Indexed / indirect
10 AC pp[1]	CMPY indexed forms	Indexed / indirect
10 AE pp[1]	LDY indexed forms	Indexed / indirect
10 AF pp[1]	STY indexed forms	Indexed / indirect
10 B3 ss qq	CMPD adr16	Extended (direct)
10 BC ss qq	CMPY adr16	Extended (direct)
10 BE ss qq	LDY adr16	Extended (direct)
10 BF ss qq	STY adr16	Extended (direct)
10 CE dd dd	LDS data16	Immediate

Note 1. The post byte may be followed by two bytes, one byte, or no byte. See Appendix B and the discussion of the post byte in Chapter 3 for more details. Appendix E lists all possible post bytes and the operand forms that produce them.

Note 2. Some instructions have two mnemonics. In each such case, we show both forms, separated by a slash (/).

Note 3. Appendix B displays the "indexed forms" for operands in the indexed and indirect addressing modes.

Note 4. In the instructions EXG and TFR, the processor interprets the second byte (the immediate data) as designating the source and destination registers.

Note 5. In the instructions PSHS, PULS, PSHU, and PULU, the processor interprets the second byte (the immediate data) as designating which registers are to be included in the transfer of data to or from the stack.

6809 Instruction Object Codes in Numerical Order (Continued)

Object Code[1]	Instruction[2, 3]	Addressing Mode
10 DE qq	LDS adr8	Base page (direct)
10 DF qq	STS adr8	Base page (direct)
10 EE pp[1]	LDS indexed forms	Indexed / indirect
10 EF pp[1]	STS indexed forms	Indexed / indirect
10 FE ss qq	LDS adr16	Extended (direct)
10 FF ss qq	STS adr16	Extended (direct)
11 3F	SWI3	Inherent
11 83 dd dd	CMPU data16	Immediate
11 8C dd dd	CMPS data16	Immediate
11 93 qq	CMPU adr8	Base page (direct)
11 9C qq	CMPS adr8	Base page (direct)
11 A3 pp[1]	CMPU indexed forms	Indexed / indirect
11 AC pp[1]	CMPS indexed forms	Indexed / indirect
11 B3 ss qq	CMPU adr16	Extended (direct)
11 BC ss qq	CMPS adr16	Extended (direct)
12	NOP	Inherent
13	SYNC	Inherent
16 mm nn	LBRA label	Relative
17 mm nn	LBSR label	Relative
19	DAA	Inherent
1A dd	ORCC data8	Immediate
1C dd	ANDCC data8	Immediate
1D	SEX	Inherent
1E dd	EXG data8	Register[4]
1F dd	TFR data8	Register[4]
20 mm	BRA label	Relative
21 mm	BRN label	Relative
22 mm	BHI label	Relative
23 mm	BLS label	Relative
24 mm	BCC label / BHS label	Relative
25 mm	BCS label / BLO label	Relative
26 mm	BNE label	Relative
27 mm	BEQ label	Relative
28 mm	BVC label	Relative
29 mm	BVS label	Relative
2A mm	BPL label	Relative
2B mm	BMI label	Relative
2C mm	BGE label	Relative
2D mm	BLT label	Relative
2E mm	BGT label	Relative
2F mm	BLE label	Relative
30 pp[1]	LEAX indexed forms	Indexed / indirect
31 pp[1]	LEAY indexed forms	Indexed / indirect
32 pp[1]	LEAS indexed forms	Indexed / indirect
33 pp[1]	LEAU indexed forms	Indexed / indirect
34 dd	PSHS data8	Register[5]
35 dd	PULS data8	Register[5]
36 dd	PSHU data8	Register[5]
37 dd	PULU data8	Register[5]
39	RTS	Inherent (Stack)
3A	ABX	Inherent
3B	RTI	Inherent (Stack)
3C dd	CWAI data8	Immediate
3D	MUL	Inherent
3F	SWI	Inherent
40	NEGA	Accumulator
43	COMA	Accumulator
44	LSRA	Accumulator
46	RORA	Accumulator

6809 Instruction Object Codes in Numerical Order (Continued)

Object Code[1]	Instruction[2, 3]	Addressing Mode
47	ASRA	Accumulator
48	ASLA / LSLA	Accumulator
49	ROLA	Accumulator
4A	DECA	Accumulator
4C	INCA	Accumulator
4D	TSTA	Accumulator
4F	CLRA	Accumulator
50	NEGB	Accumulator
53	COMB	Accumulator
54	LSRB	Accumulator
56	RORB	Accumulator
57	ASRB	Accumulator
58	ASLB / LSLB	Accumulator
59	ROLB	Accumulator
5A	DECB	Accumulator
5C	INCB	Accumulator
5D	TSTB	Accumulator
5F	CLRB	Accumulator
60 pp[1]	NEG indexed forms	Indexed / indirect
63 pp[1]	COM indexed forms	Indexed / indirect
64 pp[1]	LSR indexed forms	Indexed / indirect
66 pp[1]	ROR indexed forms	Indexed / indirect
67 pp[1]	ASR indexed forms	Indexed / indirect
68 pp[1]	ASL / LSL indexed forms	Indexed / indirect
69 pp[1]	ROL indexed forms	Indexed / indirect
6A pp[1]	DEC indexed forms	Indexed / indirect
6C pp[1]	INC indexed forms	Indexed / indirect
6D pp[1]	TST indexed forms	Indexed / indirect
6E pp[1]	JMP indexed forms	Indexed / indirect
6F pp[1]	CLR indexed forms	Indexed / indirect
70 ss qq	NEG adr16	Extended (direct)
73 ss qq	COM adr16	Extended (direct)
74 ss qq	LSR adr16	Extended (direct)
76 ss qq	ROR adr16	Extended (direct)
77 ss qq	ASR adr16	Extended (direct)
78 ss qq	ASL adr16 / LSL adr16	Extended (direct)
79 ss qq	ROL adr16	Extended (direct)
7A ss qq	DEC adr16	Extended (direct)
7C ss qq	INC adr16	Extended (direct)
7D ss qq	TST adr16	Extended (direct)
7E ss qq	JMP adr16	Extended (direct)
7F ss qq	CLR adr16	Extended (direct)
80 dd	SUBA data8	Immediate
81 dd	CMPA data8	Immediate
82 dd	SBCA data8	Immediate
83 dd dd	SUBD data16	Immediate
84 dd	ANDA data8	Immediate
85 dd	BITA data8	Immediate
86 dd	LDA data8	Immediate
88 dd	EORA data8	Immediate
89 dd	ADCA data8	Immediate
8A dd	ORA data8	Immediate
8B dd	ADDA data8	Immediate
8C dd dd	CMPX data16	Immediate
8D mm	BSR label	Relative
8E dd dd	LDX data16	Immediate
90 qq	SUBA adr8	Base page (direct)
91 qq	CMPA adr8	Base page (direct)
92 qq	SBCA adr8	Base page (direct)
93 qq	SUBD adr8	Base page (direct)

6809 Instruction Object Codes in Numerical Order (Continued)

Object Code[1]	Instruction[2, 3]	Addressing Mode
94 qq	ANDA adr8	Base page (direct)
95 qq	BITA adr8	Base page (direct)
96 qq	LDA adr8	Base page (direct)
97 qq	STA adr8	Base page (direct)
98 qq	EORA adr8	Base page (direct)
99 qq	ADCA adr8	Base page (direct)
9A qq	ORA adr8	Base page (direct)
9B qq	ADDA adr8	Base page (direct)
9C qq	CMPX adr8	Base page (direct)
9D qq	JSR adr8	Base page (direct)
9E qq	LDX adr8	Base page (direct)
9F qq	STX adr8	Base page (direct)
A0 pp[1]	SUBA indexed forms	Indexed / indirect
A1 pp[1]	CMPA indexed forms	Indexed / indirect
A2 pp[1]	SBCA indexed forms	Indexed / indirect
A3 pp[1]	SUBD indexed forms	Indexed / indirect
A4 pp[1]	ANDA indexed forms	Indexed / indirect
A5 pp[1]	BITA indexed forms	Indexed / indirect
A6 pp[1]	LDA indexed forms	Indexed / indirect
A7 pp[1]	STA indexed forms	Indexed / indirect
A8 pp[1]	EORA indexed forms	Indexed / indirect
A9 pp[1]	ADCA indexed forms	Indexed / indirect
AA pp[1]	ORA indexed forms	Indexed / indirect
AB pp[1]	ADDA indexed forms	Indexed / indirect
AC pp[1]	CMPX indexed forms	Indexed / indirect
AD pp[1]	JSR indexed forms	Indexed / indirect
AE pp[1]	LDX indexed forms	Indexed / indirect
AF pp[1]	STX indexed forms	Indexed / indirect
B0 ss qq	SUBA adr16	Extended (direct)
B1 ss qq	CMPA adr16	Extended (direct)
B2 ss qq	SBCA adr16	Extended (direct)
B3 ss qq	SUBD adr16	Extended (direct)
B4 ss qq	ANDA adr16	Extended (direct)
B5 ss qq	BITA adr16	Extended (direct)
B6 ss qq	LDA adr16	Extended (direct)
B7 ss qq	STA adr16	Extended (direct)
B8 ss qq	EORA adr16	Extended (direct)
B9 ss qq	ADCA adr16	Extended (direct)
BA ss qq	ORA adr16	Extended (direct)
BB ss qq	ADDA adr16	Extended (direct)
BC ss qq	CMPX adr16	Extended (direct)
BD ss qq	JSR adr16	Extended (direct)
BE ss qq	LDX adr16	Extended (direct)
BF ss qq	STX adr16	Extended (direct)
C0 dd	SUBB data8	Immediate
C1 dd	CMPB data8	Immediate
C2 dd	SBCB data8	Immediate
C3 dd dd	ADDD data16	Immediate
C4 dd	ANDB data8	Immediate
C5 dd	BITB data8	Immediate
C6 dd	LDB data8	Immediate
C8 dd	EORB data8	Immediate
C9 dd	ADCB data8	Immediate
CA dd	ORB data8	Immediate
CB dd	ADDB data8	Immediate
CC dd dd	LDD data16	Immediate
CE dd dd	LDU data16	Immediate
D0 qq	SUBB adr8	Base page (direct)
D1 qq	CMPB adr8	Base page (direct)

6809 Instruction Object Codes in Numerical Order (Continued)

Object Code[1]	Instruction[2, 3]	Addressing Mode
D2 qq	SBCB adr8	Base page (direct)
D3 qq	ADDD adr8	Base page (direct)
D4 qq	ANDB adr8	Base page (direct)
D5 qq	BITB adr8	Base page (direct)
D6 qq	LDB adr8	Base page (direct)
D7 qq	STB adr8	Base page (direct)
D8 qq	EORB adr8	Base page (direct)
D9 qq	ADCB adr8	Base page (direct)
DA qq	ORB adr8	Base page (direct)
DB qq	ADDB adr8	Base page (direct)
DC qq	LDD adr8	Base page (direct)
DD qq	STD adr8	Base page (direct)
DE qq	LDU adr8	Base page (direct)
DF qq	STU adr8	Base page (direct)
E0 pp[1]	SUBB indexed forms	Indexed / indirect
E1 pp[1]	CMPB indexed forms	Indexed / indirect
E2 pp[1]	SBCB indexed forms	Indexed / indirect
E3 pp[1]	ADDD indexed forms	Indexed / indirect
E4 pp[1]	ANDB indexed forms	Indexed / indirect
E5 pp[1]	BITB indexed forms	Indexed / indirect
E6 pp[1]	LDB indexed forms	Indexed / indirect
E7 pp[1]	STB indexed forms	Indexed / indirect
E8 pp[1]	EORB indexed forms	Indexed / indirect
E9 pp[1]	ADCB indexed forms	Indexed / indirect
EA pp[1]	ORB indexed forms	Indexed / indirect
EB pp[1]	ADDB indexed forms	Indexed / indirect
EC pp[1]	LDD indexed forms	Indexed / indirect
ED pp[1]	STD indexed forms	Indexed / indirect
EE pp[1]	LDU indexed forms	Indexed / indirect
EF pp[1]	STU indexed forms	Indexed / indirect
F0 ss qq	SUBB adr16	Extended (direct)
F1 ss qq	CMPB adr16	Extended (direct)
F2 ss qq	SBCB adr16	Extended (direct)
F3 ss qq	ADDD adr16	Extended (direct)
F4 ss qq	ANDB adr16	Extended (direct)
F5 ss qq	BITB adr16	Extended (direct)
F6 ss qq	LDB adr16	Extended (direct)
F7 ss qq	STB adr16	Extended (direct)
F8 ss qq	EORB adr16	Extended (direct)
F9 ss qq	ADCB adr16	Extended (direct)
FA ss qq	ORB adr16	Extended (direct)
FB ss qq	ADDB adr16	Extended (direct)
FC ss qq	LDD adr16	Extended (direct)
FD ss qq	STD adr16	Extended (direct)
FE ss qq	LDU adr16	Extended (direct)
FF ss qq	STU adr16	Extended (direct)

E

6809 Post Bytes in Numerical Order

Post Byte	Operand Form[1]	Post Byte	Operand Form[1]	Post Byte	Operand Form[1]	Post Byte	Operand Form[1]
00	0,X	37	−9,Y	6E	14,S	B4	[,Y]
01	1,X	38	−8,Y	6F	15,S	B5	[B,Y]
02	2,X	39	−7,Y	70	−16,S	B6	[A,Y]
03	3,X	3A	−6,Y	71	−15,S	B8	[nn,Y]
04	4,X	3B	−5,Y	72	−14,S	B9	[mmnn,Y]
05	5,X	3C	−4,Y	73	−13,S	BB	[D,Y]
06	6,X	3D	−3,Y	74	−12,S	BC	[nn,PC][3]
07	7,X	3E	−2,Y	75	−11,S	BD	[mmnn,PC][3]
08	8,X	3F	−1,Y	76	−10,S	BF	[mmnn]
09	9,X	40	0,U	77	−9,S	C0	,U+
0A	10,X	41	1,U	78	−8,S	C1	,U++
0B	11,X	42	2,U	79	−7,S	C2	,−U
0C	12,X	43	3,U	7A	−6,S	C3	,−−U
0D	13,X	44	4,U	7B	−5,S	C4	,U
0E	14,X	45	5,U	7C	−4,S	C5	B,U
0F	15,X	46	6,U	7D	−3,S	C6	A,U
10	−16,X	47	7,U	7E	−2,S	C8	nn,U
11	−15,X	48	8,U	7F	−1,S	C9	mmnn,U
12	−14,X	49	9,U	80	,X+	CB	D,U
13	−13,X	4A	10,U	81	,X++	CC	nn,PC[2]
14	−12,X	4B	11,U	82	,−X	CD	mmnn,PC[2]
15	−11,X	4C	12,U	83	,−−X	D1	[,U++]
16	−10,X	4D	13,U	84	,X	D3	[,−−U]
17	−9,X	4E	14,U	85	B,X	D4	[,U]
18	−8,X	4F	15,U	86	A,X	D5	[B,U]
19	−7,X	50	−16,U	88	nn,X	D6	[A,U]
1A	−6,X	51	−15,U	89	mmnn,X	D8	[nn,U]
1B	−5,X	52	−14,U	8B	D,X	D9	[mmnn,U]
1C	−4,X	53	−13,U	8C	nn,PC[2]	DB	[D,U]
1D	−3,X	54	−12,U	8D	mmnn,PC[2]	DC	[nn,PC][3]
1E	−2,X	55	−11,U	91	[,X++]	DD	[mmnn,PC][3]
1F	−1,X	56	−10,U	93	[,−−X]	DF	[mmnn]
20	0,Y	57	−9,U	94	[,X]	E0	,S+
21	1,Y	58	−8,U	95	[B,X]	E1	,S++
22	2,Y	59	−7,U	96	[A,X]	E2	,−S
23	3,Y	5A	−6,U	98	[nn,X]	E3	,−−S
24	4,Y	5B	−5,U	99	[mmnn,X]	E4	,S
25	5,Y	5C	−4,U	9B	[D,X]	E5	B,S
26	6,Y	5D	−3,U	9C	[nn,PC][3]	E6	A,S
27	7,Y	5E	−2,U	9D	[mmnn,PC][3]	E8	nn,S
28	8,Y	5F	−1,U	9F	[mmnn]	E9	mmnn,S
29	9,Y	60	0,S	A0	,Y+	EB	D,S
2A	10,Y	61	1,S	A1	,Y++	EC	nn,PC[2]
2B	11,Y	62	2,S	A2	,−Y	ED	mmnn,PC[2]
2C	12,Y	63	3,S	A3	,−−Y	F1	[,S++]
2D	13,Y	64	4,S	A4	,Y	F3	[,−−S]
2E	14,Y	65	5,S	A5	B,Y	F4	[,S]
2F	15,Y	66	6,S	A6	A,Y	F5	[B,S]
30	−16,Y	67	7,S	A8	nn,Y	F6	[A,S]
31	−15,Y	68	8,S	A9	mmnn,Y	F8	[nn,S]
32	−14,Y	69	9,S	AB	D,Y	F9	[mmnn,S]
33	−13,Y	6A	10,S	AC	nn,PC[2]	FB	[D,S]
34	−12,Y	6B	11,S	AD	mmnn,PC[2]	FC	[nn,PC][3]
35	−11,Y	6C	12,S	B1	[,Y++]	FD	[mmnn,PC][3]
36	−10,Y	6D	13,S	B3	[,−−Y]	FF	[mmnn]

Note 1: See Appendix B for addressing modes which the operand forms represent.

Note 2: May appear in source listing in the form label,PCR.

Note 3: May appear in source listing in the form [label,PCR].