Appendix B Instruction Lists

■ Explanation of symbols used in this document

1. Size

B	The operand size is in bytes (8 bits)
W	The operand size is in word (16 bits)
L	The operand size is in long word (32 bits)

2. Mnemonic

R	Eight general-purpose registers including 8/16/32-bit current bank registers. 8 bit register: W, A, B, C, D, E, H, L 16 bit register: WA, BC, DE, HL, IX, IY, IZ, SP 32 bit register: XWA, XBC, XDE, XHL, XIX, XIY, XIZ, XSP
r	8/16/32-bit general-purpose registers
cr	All 8/16/32-bit CPU control registers DMAS0 to 3, DMAD0 to 3, DMAC0 to 3, DMAM0 to 3, INTNEST
A	A register (8 bits)
F	Flag registers (8 bits)
F'	Inverse flag registers (8 bits)
SR	Status registers (16 bits)
PC	Program Counter (in minimum mode,16 bits; in maximum mode, 32 bits)
(mem)	8/16/32-bit memory data
mem	Effective address value
<W>	When the operand size is a word, "W" must be specified.
[ ]	Operands enclosed in square brackets can be omitted.
#	8/16/32-bit immediate data.
#3	3-bit immediate data: 0 to 7 or 1 to 8 ····· for abbreviated codes.
#4	4-bit immediate data: 0 to 15 or 1 to 16
d8	8-bit displacement: – 80H to + 7FH
d16	16-bit displacement: – 8000H to + 7FFFH
cc	Condition code
(#8)	Direct addressing : (00H) to (0FFH) ··· 256-byte area
(#16)	64K-byte area addressing : (0000H) to (0FFFFH)
$	A start address of the instruction is located

3. Cord

Z	The code crepresent the operand sizes. byte (8-bit) = 0 word (16-bit) = 2 long word (32-bit) = 4
ZZ	The code represent the operand sizes. byte (8-bit) = 00H word (16-bit) = 10H long word (32-bit) = 20H

4. Flag (SZHVNC)

–	Flag doesn't change.
*	Flag changes by executing instruction.
0	Flag is cleared to "0".
1	Flag is set to "1".
P	Flag changes by executing instruction (It works as parity flag).
V	Flag changes by executing instruction (It works as overflow flag).
X	An undefined value is set in flag.

5. Instruction length

Instruction length is represented in byte unit.

+#	adds immediate data length.
+M	adds addressing code length.
+#M	adds immediate data length and addressing code length.

6. State

Execution processing time of instruction are shown in order of 8 bit, 16 bit, 32 bit processing in status unit.

1 state = 100 ns at 20 MHz oscillation

1 state = 80 ns at 25 MHz oscillation

■ 900/H Instruction Lists (1/10)

(1) Load

Group	Size	Mnemonic	Codes (16 hex)	Function	SZHVNC	Length (byte)	State
LD	BWL	LD R,r	C8+zz+r :88+R	R ← r	------	2	2. 2. 2
	BWL	LD r,R	C8+zz+r :98+R	r ← R	------	2	2. 2. 2
	BWL	LD r,#3	C8+zz+r :A8+#3	r ← #3	------	2	2. 2. 2
	BWL	LD R,#	20+zz+R :#	R ← #	------	1+#	2. 3. 5
	BWL	LD r,#	C8+zz+r :03:#	r ← #	------	2+#	3. 4. 6
	BWL	LD R,(mem)	80+zz+mem:20+R	R ← (mem)	------	2+M	4. 4. 6
	BWL	LD (mem),R	B0+mem :40+zz+R	(mem) ← R	------	2+M	4. 4. 6
	BW-	LD<W> (#8) ,#	08+z :#8:#	(#8) ← #	------	2+#	5. 6. -
	BW-	LD<W> (mem),#	B0+mem :00+z:#	(mem) ← #	------	2+M#	5. 6. -
	BW-	LD<W> (#16),(mem)	80+zz+mem:19:#16	(#16) ← (mem)	------	4+M	8. 8. -
	BW-	LD<W> (mem),(#16)	B0+mem :14+z:#16	(mem) ← (#16)	------	4+M	8. 8. -
PUSH	B--	PUSH F	18	(−XSP) ← F	------	1	3. -. -
	B--	PUSH A	14	(−XSP) ← A	------	1	3. -. -
	-WL	PUSH R	18+zz+R	(−XSP) ← R	------	1	-. 3. 5
	BWL	PUSH r	C8+zz+r :04	(−XSP) ← r	------	2	4. 4. 6
	BW-	PUSH<W> #	09+z :#	(−XSP) ← #	------	1+#	4. 5. -
	BW-	PUSH<W> (mem)	80+zz+mem:04	(−XSP) ← (mem)	------	2+M	6. 6. -
POP	B--	POP F	19	F ← (XSP+)	******	1	4. -. -
	B--	POP A	15	A ← (XSP+)	------	1	4. -. -
	-WL	POP R	38+zz+R	R ← (XSP+)	------	1	-. 4. 6
	BWL	POP r	C8+zz+r :05	r ← (XSP+)	------	2	5. 5. 7
	BW-	POP<W> (mem)	B0+mem :04+z	(mem) ← (XSP+)	------	2+M	7. 7. -
LDA	-WL	LDA R,mem	B0+mem :10+zz+R	R ← mem	------	2+M	-. 4. 4
LDAR	-WL	LDAR R,$+4+d16	F3:13:d16:20+zz+R	R ← PC+d16	------	5	-. 7. 7

(2) Exchange

Group	Size	Mnemonic	Codes (16 hex)	Function	SZHVNC	Length (byte)	State
EX	B--	EX F,F'	16	F ↔ F'	******	1	2. -. -
	BW-	EX R,r	C8+zz+r :B8+R	R ↔ r	------	2	3. 3. -
	BW-	EX (mem),R	80+zz+mem:30+R	(mem) ↔ R	------	2+M	6. 6. -
MIRR	-W-	MIRR r	D8+r :16	r<0:MSB>←r<MSB:0>	------	2	-. 3. -

■ 900/H Instruction Lists (2/10)

(3) Load/Increment/Decrement & Compare Increment/Decrement Size

Group	Size	Mnemonic	Codes (16 hex)	Function	SZHVNC	Length (byte)	State
LDxx	BW-	LDI<W> [(XDE+),(XHL+)]	83+zz :10	(XDE+) ← (XHL+) BC ← BC-1	--0①0-	2	8. 8. -
	BW-	LDI<W> (XIX+),(XIY+)	85+zz :10	(XIX+) ← (XIY+) BC ← BC-1	--0①0-	2	8. 8. -
	BW-	LDIR<W> [(XDE+),(XHL+)]	83+zz :11	repeat (XDE+) ← (XHL+) BC ← BC-1 until BC=0	--000-	2	7n+1
	BW-	LDIR<W> (XIX+),(XIY+)	85+zz :11	repeat (XIX+) ← (XIY+) BC ← BC-1 until BC=0	--000-	2	7n+1
	BW-	LDD<W> [(XDE-),(XHL-)]	83+zz :12	(XDE-) ← (XHL-) BC ← BC-1	--0①0-	2	8. 8. -
	BW-	LDD<W> (XIX-),(XIY-)	85+zz :12	(XIX-) ← (XIY-) BC ← BC-1	--0①0-	2	8. 8. -
	BW-	LDDR<W> [(XDE-),(XHL-)]	83+zz :13	repeat (XDE-) ← (XHL-) BC ← BC-1 until BC=0	--000-	2	7n+1
	BW-	LDDR<W> (XIX-),(XIY-)	85+zz :13	repeat (XIX-) ← (XIY-) BC ← BC-1 until BC=0	--000-	2	7n+1
CPxx	BW-	CPI [A/WA,(R+)]	80+zz+R :14	A/WA - (R+) BC ← BC-1	*②*①1-	2	6. 6. -
	BW-	CPIR [A/WA,(R+)]	80+zz+R :15	repeat A/WA - (R+) BC ← BC-1 until A/WA=(R) or BC=0	*②*①1-	2	6n+1
	BW-	CPD [A/WA,(R-)]	80+zz+R :16	A/WA - (R-) BC ← BC-1	*②*①1-	2	6. 6. -
	BW-	CPDR [A/WA,(R-)]	80+zz+R :17	repeat A/WA - (R-) BC ← BC-1 until A/WA=(R) or BC=0	*②*①1-	2	6n+1

Note 1: ① ; If BC = 0 after execution, the P/V flag is set to 0, otherwise 1.
② ; If A/WA = (R), the Z flag is set to 1, otherwise, 0 is set.

Note 2: When the operand is omitted in the CPI, CPIR, CPD, or CPDR instruction, A, (XHL+/−) is used as the default value.

■ 900/H Instruction Lists (3/10)

(4) Arithmetic Operations

Group	Size	Mnemonic	Codes (16 hex)	Function	SZHVNC	Length (byte)	State
ADD	BWL	ADD R,r	C8+zz+r :80+R	R ← R + r	***V0*	2	2. 2. 2
	BWL	ADD r,#	C8+zz+r :C8:#	r ← r + #	***V0*	2+#	3. 4. 6
	BWL	ADD R,(mem)	80+zz+mem:80+R	R ← R + (mem)	***V0*	2+M	4. 4. 6
	BWL	ADD (mem),R	80+zz+mem:88+R	(mem) ← (mem) + R	***V0*	2+M	6. 6. 10
	BW-	ADD<W> (mem),#	80+zz+mem:38:#	(mem) ← (mem) + #	***V0*	2+M#	7. 8. -
ADC	BWL	ADC R,r	C8+zz+r :90+R	R ← R + r + CY	***V0*	2	2. 2. 2
	BWL	ADC r,#	C8+zz+r :C9:#	r ← r + # + CY	***V0*	2+#	3. 4. 6
	BWL	ADC R,(mem)	80+zz+mem:90+R	R ← R+(mem)+CY	***V0*	2+M	4. 4. 6
	BWL	ADC (mem),R	80+zz+mem:98+R	(mem) ←(mem)+ R + CY	***V0*	2+M	6. 6. 10
	BW-	ADC<W> (mem),#	80+zz+mem:39:#	(mem) ←(mem)+ # + CY	***V0*	2+M#	7. 8. -
SUB	BWL	SUB R,r	C8+zz+r :A0+R	R ← R - r	***V1*	2	2. 2. 2
	BWL	SUB r,#	C8+zz+r :CA:#	r ← r - #	***V1*	2+#	3. 4. 6
	BWL	SUB R,(mem)	80+zz+mem:A0+R	R ← R - (mem)	***V1*	2+M	4. 4. 6
	BWL	SUB (mem),R	80+zz+mem:A8+R	(mem) ← (mem) - R	***V1*	2+M	6. 6. 10
	BW-	SUB<W> (mem),#	80+zz+mem:3A:#	(mem) ← (mem) - #	***V1*	2+M#	7. 8. -
SBC	BWL	SBC R,r	C8+zz+r :B0+R	R ← R - r - CY	***V1*	2	2. 2. 2
	BWL	SBC r,#	C8+zz+r :CB:#	r ← r - # - CY	***V1*	2+#	3. 4. 6
	BWL	SBC R,(mem)	80+zz+mem:B0+R	R ←R -(mem) - CY	***V1*	2+M	4. 4. 6
	BWL	SBC (mem),R	80+zz+mem:B8+R	(mem) ←(mem) - R - CY	***V1*	2+M	6. 6. 10
	BW-	SBC<W> (mem),#	80+zz+mem:3B:#	(mem) ←(mem)- # - CY	***V1*	2+M#	7. 8. -
CP	BWL	CP R,r	C8+zz+r :F0+R	R - r	***V1*	2	2. 2. 2
	BW-	CP r,#3	C8+zz+r :D8+#3	r - #3	***V1*	2	2. 2. -
	BWL	CP r,#	C8+zz+r :CF:#	r - #	***V1*	2+#	3. 4. 6
	BWL	CP R,(mem)	80+zz+mem:F0+R	R - (mem)	***V1*	2+M	4. 4. 6
	BWL	CP (mem),R	80+zz+mem:F8+R	(mem) - R	***V1*	2+M	4. 4. 6
	BW-	CP<W> (mem),#	80+zz+mem:3F:#	(mem) - #	***V1*	2+M#	5. 6. -
INC	B--	INC #3,r	C8+r :60+#3	r ← r + #3	***V0-	2	2. -. -
	-WL	INC #3,r	C8+zz+r :60+#3	r ← r + #3	------	2	-. 2. 2
	BW-	INC<W> #3,(mem)	80+zz+mem:60+#3	(mem) ← (mem) + #3	***V0-	2+M	6. 6. -
DEC	B--	DEC #3,r	C8+r :68+#3	r ← r - #3	***V1-	2	2. -. -
	-WL	DEC #3,r	C8+zz+r :68+#3	r ← r - #3	------	2	-. 2. 2
	BW-	DEC<W> #3,(mem)	80+zz+mem:68+#3	(mem) ← (mem) - #3	***V1-	2+M	6. 6. -
NEG	BW-	NEG r	C8+zz+r :07	r ← 0 - r	***V1*	2	2. 2. -
EXTZ	-WL	EXTZ r	C8+zz+r :12	r<high> ← 0	------	2	-. 3. 3
EXTS	-WL	EXTS r	C8+zz+r :13	r<high> ← r<low. MSB>	------	2	-. 3. 3
DAA	B--	DAA r	C8+r :10	Decimal adjustment after addition or subtraction	***P-*	2	4. -. -
PAA	-WL	PAA r	C8+zz+r :14	if r<0>=1 then INC r	------	2	-. 4. 4

Note 1: With the INC/DEC instruction, when the code value of #3 = 0, functions as +8/−8.

Note 2: When the ADD R, r (word type) instruction is used in the TLCS-90, the S, Z, and V flags do not change. In the TLCS-900, these flags change.

■ 900/H Instruction Lists (4/10)

Group	Size	Mnemonic	Codes (16 hex)	Function	SZHVNC	Length (byte)	State

MUL	BW-
	BW-
	BW-
		MUL RR,r
		MUL rr,#
		MUL RR,(mem)
			C8+zz+r :40+R
			C8+zz+r :08:#
			80+zz+mem:40+R
				RR ← R×r
				rr ← r×#
				RR ← R×(mem)
					------ 	
					------ 	
					------ 	
						2
						2+#
						2+M
							11.14. -
							12.15. -
							13.16. -

MULS	BW-
	BW-
	BW-
		MULS RR,r
		MULS rr,#
		MULS RR,(mem)
			C8+zz+r :48+R
			C8+zz+r :09:#
			80+zz+mem:48+R
				RR ← R×r ;signed
				rr ← r×# ;signed
				RR ←R×(mem);signed
					------ 	
					------ 	
					------ 	
						2
						2+#
						2+M
							9.12. -
							10.13. -
							11.14. -

DIV	BW- BW- BW-	DIV RR,r DIV rr,# DIV RR,(mem)	C8+zz+r :50+R C8+zz+r :0A:# 80+zz+mem:50+R	R ← RR÷r r ← rr÷# R ← RR÷(mem)	---V-- ---V-- ---V--	2 2+# 2+M	15.23. - 15.23. - 16.24. -
DIVS	BW- BW- BW-	DIVS RR,r DIVS rr,# DIVS RR,(mem)	C8+zz+r :58+R C8+zz+r :0B:# 80+zz+mem:58+R	R ← RR÷r ;signed r ← rr÷# ;signed R ← RR÷(mem);signed	---V-- ---V-- ---V--	2 2+# 2+M	18.26. - 18.26. - 19.27. -
MULA	-W-	MULA rr	D8+r :19	Multiply and add signed rr(32bit) ← rr(32bit)+(XDE)(16bit)×(XHL)(16bit) XHL ← XHL-2	**-V--	2	-.19. -
MINC	-W-	MINC1 #,r (#=2**n) (1<=n<=15)	D8+r :38:#-1	modulo increment ;+1 if (r mod #)=(#-1) then r←r-(#-1) else r←r+1	------	4	-. 5. -
	-W-	MINC2 #,r (#=2**n) (2<=n<=15)	D8+r :39:#-2	modulo increment ;+2 if (r mod #)=(#-2) then r←r-(#-2) else r←r+2	------	4	-. 5. -
	-W-	MINC4 #,r (#=2**n) (3<=n<=15)	D8+r :3A:#-4	modulo increment ;+4 if (r mod #)=(#-4) then r←r-(#-4) else r←r+4	------	4	-. 5. -
MDEC	-W-	MDEC1 #,r (#=2**n) (1<=n<=15)	D8+r :3C:#-1	modulo decrement ;-1 if (r mod #)=0 then r←r+(#-1) else r←r-1	------	4	-. 4. -
	-W-	MDEC2 #,r (#=2**n) (2<=n<=15)	D8+r :3D:#-2	modulo decrement ;-2 if (r mod #)=0 then r←r+(#-2) else r←r-2	------	4	-. 4. -
	-W-	MDEC4 #,r (#=2**n) (3<=n<=15)	D8+r :3E:#-4	modulo decrement ;-4 if (r mod #)=0 then r←r+(#-4) else r←r-4	------	4	-. 4. -

Note: Operand RR of the MUL, MULS, DIV, and DIVS instructions indicates that a register twice the size of the operation is specified. When the operation is in bytes (8 bits×8 bits, 16/8 bits), word register (16 bits) is specified; when the operation is in words (16 bits×16 bits, 32/16 bits), long word register (32 bits) is specified.

■ 900/H Instruction Lists (5/10)

(5) Logical operations

Group	Size	Mnemonic	Codes (16 hex)	Function	SZHVNC	Length (byte)	State
AND	BWL	AND R,r	C8+zz+r :C0+R	R ← R and r	**1P00	2	2. 2. 2
	BWL	AND r,#	C8+zz+r :CC:#	r ← r and #	**1P00	2+#	3. 4. 6
	BWL	AND R,(mem)	80+zz+mem:C0+R	R ← R and (mem)	**1P00	2+M	4. 4. 6
	BWL	AND (mem),R	80+zz+mem:C8+R	(mem) ← (mem) and R	**1P00	2+M	6. 6. 10
	BW-	AND<w> (mem),#	80+zz+mem:3C:#	(mem) ← (mem) and #	**1P00	2+M#	7. 8. -
OR	BWL	OR R,r	C8+zz+r :E0+R	R ← R or r	**0P00	2	2. 2. 2
	BWL	OR r,#	C8+zz+r :CE:#	r ← r or #	**0P00	2+#	3. 4. 6
	BWL	OR R,(mem)	80+zz+mem:E0+R	R ← R or (mem)	**0P00	2+M	4. 4. 6
	BWL	OR (mem),R	80+zz+mem:E8+R	(mem) ← (mem) or R	**0P00	2+M	6. 6. 10
	BW-	OR<W> (mem),#	80+zz+mem:3E:#	(mem) ← (mem) or #	**0P00	2+M#	7. 8. -
XOR	BWL	XOR R,r	C8+zz+r :D0+R	R ← R xor r	**0P00	2	2. 2. 2
	BWL	XOR r,#	C8+zz+r :CD:#	r ← r xor #	**0P00	2+#	3. 4. 6
	BWL	XOR R,(mem)	80+zz+mem:D0+R	R ← R xor (mem)	**0P00	2+M	4. 4. 6
	BWL	XOR (mem),R	80+zz+mem:D8+R	(mem) ← (mem) xor R	**0P00	2+M	6. 6. 10
	BW-	XOR<W> (mem),#	80+zz+mem:3D:#	(mem) ← (mem) xor #	**0P00	2+M#	7. 8. -
CPL	BW-	CPL r	C8+zz+r :06	r ← not r	--1-1-	2	2. 2. -

■ 900/H Instruction Lists (6/10)

(6) Bit operations

Group	Size	Mnemonic	Codes (16 hex)	Function	SZHVNC	Length (byte)	State

LDCF	BW-
	BW-
	B--
	B--
		LDCF #4,r
		LDCF A ,r
		LDCF #3,(mem)
		LDCF A ,(mem)
			C8+zz+r :23:#4
			C8+zz+r :2B
			B0+men :98+#3
			B0+men :2B
				CY ← r<#4>
				CY ← r<A>
				CY ← (mem)<#3>
				CY ← (mem)<A>
					-----* 	
					-----* 	
					-----* 	
					-----* 	
						3
						2
						2+M
						2+M
							3.3.-
							3.3.-
							6.-.-
							6.-.-

STCF	BW- BW- B-- B--	STCF #4,r STCF A ,r STCF #3,(mem) STCF A ,(mem)	C8+zz+r :24:#4 C8+zz+r :2C B0+mem :A0+#3 B0+mem :2C	r<#4> ← CY r<A> ← CY (mem)<#3> ← CY (mem)<A> ← CY	------ ------ ------ ------	3 2 2+M 2+M	3.3.- 3.3.- 7.-.- 7.-.-
ANDCF	BW- BW- B-- B--	ANDCF #4,r ANDCF A ,r ANDCF #3,(mem) ANDCF A ,(mem)	C8+zz+r :20:#4 C8+zz+r :28 B0+mem :80+#3 B0+mem :28	CY ← CY and r<#4> CY ← CY and r<A> CY ← CY and (mem)<#3> CY ← CY and (mem)<A>	-----* -----* -----* -----*	3 2 2+M 2+M	3.3.- 3.3.- 6.-.- 6.-.-
ORCF	BW- BW- B-- B--	ORCF #4,r ORCF A ,r ORCF #3,(mem) ORCF A ,(mem)	C8+zz+r :21:#4 C8+zz+r :29 B0+mem :88+#3 B0+mem :29	CY ← CY or r<#4> CY ← CY or r<A> CY ← CY or (mem)<#3> CY ← CY or (mem)<A>	-----* -----* -----* -----*	3 2 2+M 2+M	3.3.- 3.3.- 6.-.- 6.-.-
XORCF	BW- BW- B-- B--	XORCF #4,r XORCF A ,r XORCF #3,(mem) XORCF A ,(mem)	C8+zz+r :22:#4 C8+zz+r :2A B0+mem :90+#3 B0+mem :2A	CY ← CY xor r<#4> CY ← CY xor r<A> CY ← CY xor (mem)<#3> CY ← CY xor (mem)<A>	-----* -----* -----* -----*	3 2 2+M 2+M	3.3.- 3.3.- 6.-.- 6.-.-
RCF SCF CCF ZCF	--- --- --- ---	RCF SCF CCF ZCF	10 11 12 13	CY ← 0 CY ← 1 CY ← not CY CY ← not "Z" flag	--0-00 --0-01 --X-0* --X-0*	1 1 1 1	2 2 2 2
BIT	BW- B--	BIT #4,r BIT #3,(mem)	C8+zz+r :33:#4 B0+mem :C8+#3	Z ← not r<#4> Z ← not (mem)<#3>	X*1X0- X*1X0-	3 2+M	3.3.- 6.-.-
RES	BW- B--	RES #4,r RES #3,(mem)	C8+zz+r :30:#4 B0+mem :B0+#3	r<#4> ← 0 (mem)<#3> ← 0	------ ------	3 2+M	3.3.- 7.-.-
SET	BW- B--	SET #4,r SET #3,(mem)	C8+zz+r :31:#4 B0+mem :B8+#3	r<#4> ← 1 (mem)<#3> ← 1	------ ------	3 2+M	3.3.- 7.-.-
CHG	BW- B--	CHG #4,r CHG #3,(mem)	C8+zz+r :32:#4 B0+mem :C0+#3	r<#4> ← not r<#4> (mem)<#3>←not (mem)<#3>	------ ------	3 2+M	3.3.- 7.-.-
TSET	BW- B--	TSET #4,r TSET #3,(mem)	C8+zz+r :34:#4 B0+mem :A8+#3	Z←not r<#4> : r<#4>←1 Z ← not (mem)<#3> (mem)<#3> ← 1	X*1X0- X*1X0-	3 2+M	4.4.- 7.-.-
BS1	-W- -W-	BS1F A,r BS1B A,r	D8+r :0E D8+r :0F	A ← 1 search r ; Forward A ← 1 search r ; Backward	---①-- ---①--	2 2	-.3.- -.3.-

Note: ① ; 0 is set when the bit searched for is found, otherwise 1 is set and an undefined value is set in the A register.

■ 900/H Instruction Lists (7/10)

(7) Special operations and CPU control

Group	Size	Mnemonic	Codes (16 hex)	Function	SZHVNC	Length (byte)	State
NOP	---	NOP	00	no operation	------	1	2
EI	---	EI [#3]	06 :#3	Sets interrupt enable flag. IFF←#3	------	2	3
DI	---	DI	06 :07	Disables interrupt. IFF←7	------	2	4
PUSH	-W-	PUSH SR	02	(-XSP)←SR	------	1	-.3.-
POP	-W-	POP SR	03	SR←(XSP+)	******	1	-.4.-
SWI	---	SWI [#3]	F8+#3	Software interrupt PUSH PC&SR JP (FFFF00H+4×#3)	------	1	19
HALT	---	HALT	05	CPU halt	------	1	6
LDC	BWL BWL	LDC cr,r LDC r,cr	C8+zz+r :2E:cr C8+zz+r :2F:cr	cr ← r r ← cr	------ ------	3 3	3.3.3 3.3.3
LDX	B--	LDX (#8),#	F7:00:#8:00:#:00	(#8) ← #	------	6	8.-.-

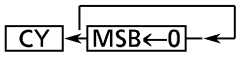
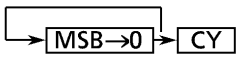
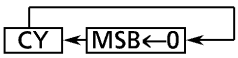
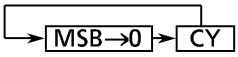
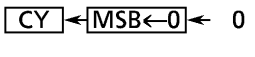
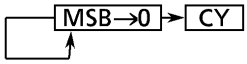
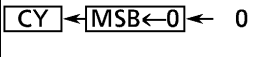
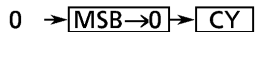
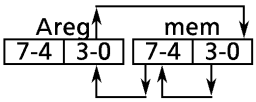
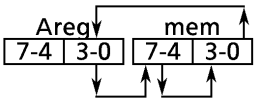
LINK	--L	LINK r,d16	E8+r :0C:d16	PUSH r
				LD r,XSP
				ADD XSP,d16
					------	4	-.-.8

UNLK	--L	UNLK r	E8+r :0D	LD XSP,r POP r	------	2	-.-.7
LDF	---	LDF #3	17 :#3	Seta register bank. RFP ← #3 (0 at reset)	------	2	2
INCF	---	INCF	0C	Switches register banks. RFP← RFP + 1	------	1	2
DECF	---	DECF	0D	Switches register banks. RFP← RFP – 1	------	1	2
SCC	BW-	SCC cc,r	C8+zz+r :70+cc	if cc then r ← 1 else r ← 0	------	2	2.2.-

Note 1: When operand #3 coding in the EI instruction is omitted, 0 is used as the default value.
Note 2: When operand #3 coding in the SWI instruction is omitted, 7 is used as the default value.

■ 900/H Instruction Lists (8/10)

(8) Rotate and Shift

Group	Size	Mnemonic	Codes (16 hex)	Function	SZHVNC	Length (byte)	State
RLC	BWL	RLC #4,r	C8+zz+r :E8:#4		**0P0*	3	3+n/4
	BWL	RLC A,r	C8+zz+r :F8		**0P0*	2	3+n/4
	BW-	RLC<W> (mem)	80+zz+mem:78		**0P0*	2+M	6.6
RRC	BWL	RRC #4,r	C8+zz+r :E9:#4		**0P0*	3	3+n/4
	BWL	RRC A,r	C8+zz+r :F9		**0P0*	2	3+n/4
	BW-	RRC<W> (mem)	80+zz+mem:79		**0P0*	2+M	6.6
RL	BWL	RL #4,r	C8+zz+r :EA:#4		**0P0*	3	3+n/4
	BWL	RL A,r	C8+zz+r :FA		**0P0*	2	3+n/4
	BW-	RL<W> (mem)	80+zz+mem:7A		**0P0*	2+M	6.6
RR	BWL	RR #4,r	C8+zz+r :EB:#4		**0P0*	3	3+n/4
	BWL	RR A,r	C8+zz+r :FB		**0P0*	2	3+n/4
	BW-	RR<W> (mem)	80+zz+mem:7B		**0P0*	2+M	6.6
SLA	BWL	SLA #4,r	C8+zz+r :EC:#4		**0P0*	3	3+n/4
	BWL	SLA A,r	C8+zz+r :FC		**0P0*	2	3+n/4
	BW-	SLA<W> (mem)	80+zz+mem:7C		**0P0*	2+M	6.6
SRA	BWL	SRA #4,r	C8+zz+r :ED:#4		**0P0*	3	3+n/4
	BWL	SRA A,r	C8+zz+r :FD		**0P0*	2	3+n/4
	BW-	SRA<W> (mem)	80+zz+mem:7D		**0P0*	2+M	6.6
SLL	BWL	SLL #4,r	C8+zz+r :EE:#4		**0P0*	3	3+n/4
	BWL	SLL A,r	C8+zz+r :FE		**0P0*	2	3+n/4
	BW-	SLL<W> (mem)	80+zz+mem:7E		**0P0*	2+M	6.6
SRL	BWL	SRL #4,r	C8+zz+r :EF:#4		**0P0*	3	3+n/4
	BWL	SRL A,r	C8+zz+r :FF		**0P0*	2	3+n/4
	BW-	SRL<W> (mem)	80+zz+mem:7F		**0P0*	2+M	6.6
RLD	B--	RLD [A,](mem)	80+mem :06		**0P0-	2+M	14.-.-
RRD	B--	RRD [A,](mem)	80+mem :07		**0P0-	2+M	14.-.-

Note 1: When #4/A is used to specify the number of shifts, module 16 (0 to 15) is used. Code 0 means 16 shifts.

Note 2: When the following instructions are used in the TLCS-90, the S, Z and V flags do not change.

Note 3: RLCA, RRCA, RLA, RRA, SLAA, SRAA, SLLA, and SRLA
In the TLCS-900, these flags change.

■ 900/H Instruction Lists (9/10)

(9) Jump, Call and Return

Group	Size	Mnemonic	Codes (16 hex)	Function	SZHVNC	Length (byte)	State
JP	---	JP #16	1A :#16	PC ← #16	------	3	5
	---	JP #24	1B :#24	PC ← #24	------	4	6
	---	JR [cc,]$+2+d8	60+cc :d8	if cc then PC ← PC+d8	------	2	5/2 (T/F)
	---	JRL [cc,]$+3+d16	70+cc :d16	if cc then PC ← PC+d16	------	3	5/2 (T/F)
	---	JP [cc,]mem	B0+mem :D0+cc	if cc then PC ← mem	------	2+M	7/4 (T/F)
CALL	---	CALL #16	1C :#16	PUSH PC : JP #16	------	3	9
	---	CALL #24	1D :#24	PUSH PC : JP #24	------	4	10
	---	CALR $+3+d16	1E :d16	PUSH PC : JR $+3+d16	------	3	10
	---	CALL [cc,]mem	B0+mem :E0+cc	if cc then PUSH PC : JP mem	------	2+M	12/4 (T/F)
DJNZ	BW-	DJNZ [r,]$+3/4+d8	C8+zz+r :1C:d8	r←r-1 if r≠0 then JR $+3+d8	------	3	6 (r≠0) 4 (r=0)
RET	---	RET	0E	POP PC	------	1	9
	---	RET cc	B0 :F0+cc	if cc then POP PC	------	2	12/4 (T/F)
	---	RETD d16	0F :d16	RET : ADD XSP,d16	------	3	11
	---	RETI	07	POP SR&PC	******	1	12

Note 1: (T/F) represents the number of states at true / false.

■ Instruction Lists of 900/H (10/10)

(10) Addressing mode

type	mode	state
R	R	+0
r	r	+1
(mem)	(R)	+0
	(R + d8)	+1
	(#8)	+1
	(#16)	+2
	(#24)	+3
	(r)	+1
	(r + d16)	+3
	(r + r8)	+3
	(r + r16)	+3
	(– r)	+1
	(r +)	+1

(11) Interrupt

mode		operation	state
General-purpose interrupt processing		PUSH PC PUSH SR IFF ← accepted level + 1 INTNEST ← INTNEST + 1 JP (FFFF00H + vector)	18
HDMA	I/O to MEM	(DMADn +) ← (DMASn)	8. 8. 12
	I/O to MEM	(DMADn –) ← (DMASn)	8. 8. 12
	MEM to I/O	(DMADn) ← (DMASn +)	8. 8. 12
	MEM to I/O	(DMADn) ← (DMASn –)	8. 8. 12
	I/O to I/O	(DMADn) ← (DMASn)	8. 8. 12
	Counter	DMASn ← DMASn + 1	– . – . 5

(Note) For details of interrupt processing, refer to Chapter2 "3.3 Interrupts".

Appendix C Instruction Code Maps (1/4)

1-byte op code instructions

H/L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP		PUSH SR	POP SR		HALT	EI n	RETI	LD (n) , n	PUSH n	LDW (n) , nn	PUSHW nn	INCF	DECF	RET	RETD dd
1	RCF	SCF	CCF	ZCF	PUSH A	POP A	EX F, F'	LDF n	PUSH F	POP F	JP nn	JP nnn	CALL nn	CALL nnn	CALR PC + dd	
2	LD R, n								PUSH RR							
3	LD RR, nn								PUSH XRR							
4	LD XRR, nnnn								POP RR							
5									POP XRR							
6	F	LT	LE	ULE	PE/OV	M/MI	Z	JR C	cc,PC + d (T)	GE	GT	UGT	PO/NOV	P/PL	NZ	NC
7	F	LT	LE	ULE	PE/OV	M/MI	Z	JRL C	cc,PC + dd (T)	GE	GT	UGT	PO/NOV	P/PL	NZ	NC
8	(XWA)	(XBC)	(XDE)	(XHL)	(XIX)	(XIY)	(XIZ)	(XSP)	(XWA + d)	(XBC + d)	(XDE + d)	scr. B (XHL + d)	(XIX + d)	(XIY + d)	(XIZ + d)	(XSP + d)
9	(XWA)	(XBC)	(XDE)	(XHL)	(XIX)	(XIY)	(XIZ)	(XSP)	(XWA + d)	(XBC + d)	(XDE + d)	scr. W (XHL + d)	(XIX + d)	(XIY + d)	(XIZ + d)	(XSP + d)
A	(XWA)	(XBC)	(XDE)	(XHL)	(XIX)	(XIY)	(XIZ)	(XSP)	(XWA + d)	(XBC + d)	(XDE + d)	scr. L (XHL + d)	(XIX + d)	(XIY + d)	(XIZ + d)	(XSP + d)
B	(XWA)	(XBC)	(XDE)	(XHL)	(XIX)	(XIY)	(XIZ)	(XSP)	(XWA + d)	(XBC + d)	(XDE + d)	dst (XHL + d)	(XIX + d)	(XIY + d)	(XIZ + d)	(XSP + d)
C	scr. B (n)	(nn)	(nnn)	(mem)	(– xrr)	(xrr +)		reg. B r	W	A	B	reg. B C	D	E	H	L
D	scr. W (n)	(nn)	(nnn)	(mem)	(– xrr)	(xrr +)		reg. W rr	WA	BC	DE	reg. W HL	IX	IY	IZ	SP
E	scr. L (n)	(nn)	(nnn)	(mem)	(– xrr)	(xrr +)		reg. L xrr	XWA	XBC	XDE	reg. L XHL	XIX	XIY	XIZ	XSP
F	dst (n)	(nn)	(nnn)	(mem)	(– xrr)	(xrr +)		LDX (n), n	0	1	2	SWI 3	n 4	5	6	7

Note 1: Codes in blank parts are undefined instructions (i.e., illegal instructions).
Note 2: Dummy instructions are assigned to codes 01H and 04H. Do not use them.

Appendix C Instruction Code Maps (2/4)

1st byte: reg

H/L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				LD r,#	PUSH r	POP r	CPL BW r	NEG BW r	MUL rr,#	MULS rr,#	DIV rr,#	DIVS BW rr,#	LINK L r, dd	UNLK L r	BS1F A, r	BS1B W A, r
1	DAA B r		EXTZ WL r	EXTS WL r	PAA WL r		MIRR W r			MULA W r			DJNZ BW r, d			
2	ANDCF #, r	ORCF #, r	XORCF #, r	LDCF #, r	STCF BW #, r				ANDCF A, r	ORCF A, r	XORCF A, r	LDCF A, r	STCF BW A, r		LDC cr, r	LDC r, cr
3	RES #, r	SET #, r	CHG #, r	BIT #, r	TSET BW #, r				MINC1	MINC2 #, r	MINC4 W		MDEC1	MDEC2 #, r	MDEC4 W	
4				MUL	R, r			BW				MULS	R, r			BW
5				DIV	R, r			BW				DIVS	R, r			BW
6	8	1	2	INC 3	#3, r 4	5	6	7	8	1	2	DEC 3	#3, r 4	5	6	7
7	F	LT	LE	ULE	PE/OV	M/MI	Z	SCC C	cc, r (T)	GE	GT	UGT	PO/NOV	P/PL	NZ	BW NC
8				ADD	R, r							LD	R, r			
9				ADC	R, r							LD	r, R			
A				SUB	R, r				0	1	2	LD 3	r, #3 4	5	6	7
B				SBC	R, r							EX	R, r			BW
C				AND	R, r				ADD r, #	ADC r, #	SUB r, #	SBC r, #	AND r, #	XOR r, #	OR r, #	CP r, #
D				XOR	R, r				0	1	2	CP 3	r, #3 4	5	6	BW 7
E				OR	R, r				RLC #, r	RRC #, r	RL #, r	RR #, r	SLA #, r	SRA #, r	SLL #, r	SRL #, r
F				CP	R, r				RLC A, r	RRC A, r	RL A, r	RR A, r	SLA A, r	SRA A, r	SLL A, r	SRL A, r

r : Register specified by the 1st byte code. (Any CPU registers can be specified.)
R : Register specified by the 2nd byte code. (Only eight current registers can be specified.)
B : Operand size is a byte.
W : Operand size is a word.
L : Operand size is a long word.

Note : Dummy instructions are assigned to codes 1AH, 1BH, 3BH, and 3FH. Do not use them.

Appendix C Instruction Code Maps (3/4)

1st byte: src (mem)

H/L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0					PUSH BW (mem)		RLD A, (mem)	RLD B								
1	LDI	LDIR	LDD	LDDR BW	CPI	CPIR	CPD	CPDR BW		LD BW (nn), (m)						
2	LD R, (mem)															
3	EX (mem), R BW								ADD (mem) ,#	ADC	SUB	SBC	AND	XOR	OR	CP BW
4	MUL R, (mem) BW								MULS R, (mem) BW							
5	DIV R, (mem) BW								DIVS R, (mem) BW							
6	INC #3, (mem) BW								DEC #3, (mem) BW							
	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7
7									RLC (mem)	RRC	RL	RR	SLA	SRA	SLL	SRL BW
8	ADD R, (mem)								ADD (mem), R							
9	ADC R, (mem)								ADC (mem), R							
A	SUB R, (mem)								SUB (mem), R							
B	SBC R, (mem)								SBC (mem), R							
C	AND R, (mem)								AND (mem), R							
D	XOR R, (mem)								XOR (mem), R							
E	OR R, (mem)								OR (mem), R							
F	CP R, (mem)								CP (mem), R							

B : Operand size is a byte.
W : Operand size is a word.

Appendix C Instruction Code Maps (4/4)

1st byte: dst (mem)

H/L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	LD B (m), #		LD W (m), #		POP B (mem)		POP W (mem)									
1					LD B (m), (nn)		LD W (m), (nn)									
2	LDA R, mem W								ANDCF A, (mem)	ORCF	XORCF	LDCF	STCF B			
3	LDA R, mem L															
4	LD (mem), R B															
5	LD (mem), R W															
6	LD (mem), R L															
7																
8	ANDCF #3, (mem) B 0	1	2	3	4	5	6	7	0	1	2	ORCF #3, (mem) B 3	4	5	6	7
9	XORCF #3, (mem) B 0	1	2	3	4	5	6	7	0	1	2	LDCF #3, (mem) B 3	4	5	6	7
A	STCF #3, (mem) B 0	1	2	3	4	5	6	7	0	1	2	TSET #3, (mem) B 3	4	5	6	7
B	RES #3, (mem) B 0	1	2	3	4	5	6	7	0	1	2	SET #3, (mem) B 3	4	5	6	7
C	CHG #3, (mem) B 0	1	2	3	4	5	6	7	0	1	2	BIT #3, (mem) B 3	4	5	6	7
D	F	LT	LE	ULE	PE/OV	M/MI	Z	JP cc, mem C	(T)	GE	GT	UGT	PO/NOV	P/PL	NZ	NC
E	F	LT	LE	ULE	PE/OV	M/MI	Z	CALL cc, mem C	(T)	GE	GT	UGT	PO/NOV	P/PL	NZ	NC
F	F	LT	LE	ULE	PE/OV	M/MI	Z	RET cc C	(T)	(1st byte code is B0H.) GE	GT	UGT	PO/NOV	P/PL	NZ	NC

B : Operand size is a byte.
W : Operand size is a word.
L : Operand size is a long word.

Appendix D Differences between TLCS-90 and TLCS-900 Series

Item \ Series	TLCS-90	TLCS-900
CPU architecture	8-bit CPU	16-bit CPU
Built-in ROM/built-in RAM	8-bit data bus	16-bit data bus
Built-in I/O	8-bit data bus	8-bit data bus
External data bus	8-bit data bus	8-bit/16-bit data bus (can be mixed)
Program space (except devices with MMU)	64 KB	16MB (linear)
Data space	16MB (bank)	16MB (linear)
Instruction set/instruction mnemonic	TLCS-90	TLCS-90 + α α = enhancement of 16-bit multiply / divide instructions and bit operation instruction. 32-bit load/operation instructions, C compiler instructions, register bank operation instructions, etc.
Instruction code (object code)	Unique to TLCS-90	Unique to TLCS-900 (Different from TLCS-90.)
Addressing mode	TLCS-90	TLCS-90 + α α = (– Reg), (Reg +), (Reg + disp16), (Reg + Reg16), (nnn)
General-purpose register	TLCS-90	TLCS-90 + α α = Uses as 32 bits and register bank, and adds a system stack pointer.
Flag (F)	S Z I H X V N C	S Z "0" H "0" V N C I flag is extended to IFF2 to 0 of status register.X flag is deleted.
Reset	PC ← 0000H (SP does not change.)	PC ← (Vector base address) XSP ← 100H
Built-in ROM address Built-in RAM address Built-in I/O address Direct addressing area (n)	0000H to to FFxxH FFxxH~FFFFH FF00H~FFFFH	undefined 0080H to 0000H~007FH 0000H~00FFH
Interrupt Interrupt start address Register to be saved Mask register Mask level	 0000H + (8 × V) PC & AF IFF 0~1	 Vector base address + 4 × V PC & SR IFF2~0 0~7

Item \ Series	TLCS-90	TLCS-900
Instruction		
① ADD R, r (word type)	S/Z/V flags don't change. [S/Z/V flag changes expect add 16 bit register.]	S/Z/V flag changes.
② Shift of A register	RLCA RRCA RLA RRA SLAA SRAA SLLA SRLA S/Z/V flags don't change in these instruction. RLC A RRC A RL A RR A SLA A SRA A SLL A SRL A S/Z/V flag changes in these instruction.	S/Z/V flag changes.

Note : The TLCS-900 series is essentially the same as the TLCS-90 series but with a 16-bit CPU. Built-in I/Os are completely compatible with those of the TLCS-90.
However, six types of instructions used in the TLCS-90 series do not directly correspond with those used in the TLCS-900 series. Thus, when transfering programs designed for the TLCS-90 to the TLCS-900, replace them with equivalents as follows:

Instructions in TLCS-90 but not in TLCS-900	Equivalent instructions in TLCS-900
EXX	EX BC, BC' EX DE, DE' EX HL, HL'
EX AF, AF'	EX A, A' EX F, F'
PUSH AF	PUSH A PUSH F
POP AF	POP F POP A
INCX	(32-bit INC instruction)
DECX	(32-bit DEC instruction)

Some TLCS-900 series instructions, though basically the same as TLCS-90 instructions, have more functions and more specification items in their operands. They are listed below.

TLCS-90	TLCS-900
INC reg INC mem	INC imm3, reg INC imm3, mem
DEC reg DEC mem	DEC imm3, reg DEC imm3, mem
RLC reg RRC reg RL reg RR reg SLA reg SRA reg SLL reg SRL reg	RLC imm, reg RRC imm, reg RL imm, reg RR imm, reg SLA imm, reg SRA imm, reg SLL imm, reg SRL imm, reg