5. Instructions

In addition to its various addressing modes, the TLCS-900 series also has a powerful instruction set. The basic instructions are classified into the following nine groups:

- Load instructions (8/16/32 bits)
- Exchange instructions (8/16 bits)
- Block transfer and Block search instructions (8/16 bits)
- Arithmetic operation instructions (8/16/32 bits)
- Logical operation instructions (8/16/32 bits)
- Bit operation instructions (1 bit)
- Special operations, CPU control instructions
- Rotate and Shift instructions (8/16/32 bits)
- Jump, Call, and Return instructions

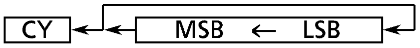
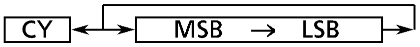
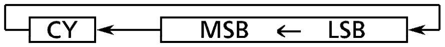
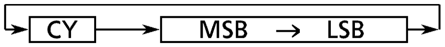
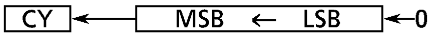
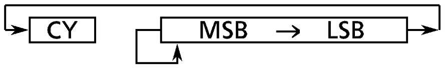
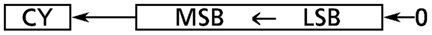
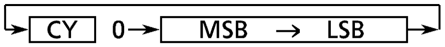
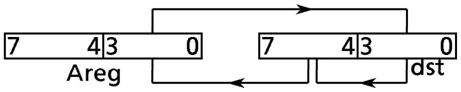
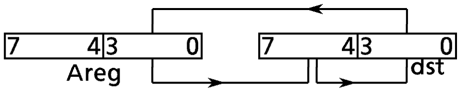
Table 5.1 lists the basic instructions of the TLCS-900 series. For details of instructions, see Appendix A; for the instruction list, Appendix B; for the instruction code map, Appendix C; and for the differences between the TLCS-90 and TLCS-900 series, Appendix D.

Table 5.1 TLCS-900 Series Basic Instructions

LD	dst, src	Load dst←src	
PUSH	src	Push src data to stack. SP←SP – size: (SP) ←src	
POP	dst	Pop data from stack to dst. dst←(SP) : SP←SP + size	
LDA	dst, src	Load address: set src effective address in dst.	
LDAR	dst, PC + dd	Load address relative: set program counter relative address value in dst. dst←PC + dd	
EX	dst1, dst2	Exchange dst1 and dst2 data.	
MIRR	dst	Mirror-invert dst bit pattern.	
LDI		Load increment	
LDIR		Load increment repeat	
LDD		Load decrement	
LDDR		Load decrement repeat	
CPI		Compare increment	
CPIR		Compare increment repeat	
CPD		Compare decrement	
CPDR		Compare decrement repeat	
ADD	dst, src	Add	dst←dst + src
ADC	dst, src	Add with carry	dst←dst + src + CY
SUB	dst, src	Subtract	dst←dst – src
SBC	dst, src	Subtract with carry	dst←dst – src – CY
CP	dst, src	Compare	dst – src
AND	dst, src	And	dst←dst AND src
OR	dst, src	Or	dst←dst OR src
XOR	dst, src	Exclusive-or	dst←dst XOR src
INC	imm, dst	Increment	dst←dst + imm
DEC	imm, dst	Decrement	dst←dst – imm
MUL	dst, src	Multiply unsigned	dst←dst (low) × src
MULS	dst, src	Multiply signed	dst←dst (low) × src
DIV	dst, src	Divide unsigned dst (low) ← dst ÷ src dst (high) ← remainder V flag set due to division by 0 or overflow.	
DIVS	dst, src	Divide signed dst (low) ← dst ÷ src dst (high) ← remainder: sign is same as that of dividend. V flag set due to division by 0 or overflow.	

MULA	dst	Multiply and add $\underset{\text{32bits}}{\underline{\text{dst}}} \leftarrow \underset{\text{32bits}}{\underline{\text{dst}}} + \underset{\text{16bits}}{\underline{\text{(XDE)}}} \times \underset{\text{16bits}}{\underline{\text{(XHL}-\text{)}}}$
MINC1	num, dst	Modulo increment 1
MINC2	num, dst	Modulo increment 2
MINC4	num, dst	Modulo increment 4
MDEC1	num, dst	Modulo decrement 1
MDEC2	num, dst	Modulo decrement 2
MDEC4	num, dst	Modulo decrement 4
NEG	dst	Negate dst←0 – dst (Twos complement)
CPL	dst	Complement dst←not dst (Ones complement)
EXTZ	dst	Extend zero: set upper data of dst to 0.
EXTS	dst	Extend signed: copy the MSB of the lower data of dst to upper data.
DAA	dst	Decimal adjustment accumulator
PAA	dst	Pointer adjustment accumulator: when dst is odd, increment dst by 1 to make it even. if dst (0) = 1 then dst←dst + 1.
LDCF	bit, src	Load carry flag: copy src<bit> value to C flag.
STCF	bit, dst	Store carry flag: copy C flag value to dst<bit>.
ANDCF	bit, src	And carry flag: and src<bit> value and C flag, then load the result to C flag.
ORCF	bit, src	Or carry flag: or src<bit> and C flag, then load result to C flag.
XORCF	bit, src	Exclusive-or carry flag: exclusive-or src<bit> value and C flag, then load result to C flag.
RCF		Reset carry flag: reset C flag to 0.
SCF		Set carry flag: set C flag to 1.
CCF		Complement carry flag: invert C flag value.
ZCF		Zero flag to carry flag: copy inverted value of Z flag to C flag.
BIT	bit, src	Bit test: Z flag ← not src<bit>
RES	bit, dst	Bit reset dst<bit> ← 0
SET	bit, dst	Bit set dst<bit> ← 1
CHG	bit, dst	Bit change dst<bit>←not dst<bit>
TSET	bit, dst	Bit test and set: Z flag ← not dst<bit> dst<bit> ← 1

BS1F	A, dst	Bit search 1 forward: search dst for the first bit set to 1 starting from the LSB, then set the bit number in the A register.
BS1B	A,dst	Bit search 1 backward: search dst for the first bit set to 1 starting fom the MSB, then set the bit number in the A register.
NOP		No operation
EI	imm	Enable interrupt. IFF←imm
DI		Disable maskable interrupt. IFF←7
PUSH	SR	Push status registers.
POP	SR	Pop status registers.
SWI	imm	Software interrupt PUSH PC&SR JP FFFF00H + 10H × imm
HALT		Halt CPU.
LDC	CTRL – REG, reg	Load control: copy the register contents to control register of CPU.
LDC	reg, CTRL – REG	Load control: copy the control register contents to register.
LDX	dst, src	Load extract. dst←src
LINK	reg, dd	Link: generate stack frame. PUSH reg LD reg, XSP ADD XSP, dd
UNLK	reg	Unlink: delete stack frame. LD XSP, reg POP reg
LDF	imm	Load register file pointer: specify register bank. RFP←imm
INCF		Increment register file pointer: move to new register bank. RFP←RFP + 1
DECF		Decrement register file pointer: return to previous register bank. RFP←RFP – 1
SCC	cc, dst	Set dst with condition codes. if cc then dst ←1 else dst ←0.

RLC	num, dst	Rotate left without carry	
RRC	num, dst	Rotate right without carry	
RL	num, dst	Rotate left	
RR	num, dst	Rotate right	
SLA	num, dst	Shift left arithmetic	
SRA	num, dst	Shift right arithmetic	
SLL	num, dst	Shift left logical	
SRL	num, dst	Shift right logical	
RLD	dst	Rotate left digit	
RRD	dst	Rotate right digit	
JR	cc, PC + d	Jump relative (8-bit displacement) if cc then PC←PC + d.	
JRL	cc, PC + dd	Jump relative long (16-bit displacement) if cc then PC←PC + dd.	
JP	cc, dst	Jump if cc then PC←dst.	
CALR	RC + dd	Relative call (16-bit displacement) PUSH PC PC←PC + dd.	

CALL	cc, dst	Call relative
		if cc then PUSH PC
		PC←dst.	

DJNZ	dst, PC + d	Decrement and jump if non-zero
		dst←dst − 1
		if dst ≠ 0 then PC←PC + d.	

RET	cc	Return if cc then POP PC.	
RETD	dd	Return and deallocate RET XSP←XSP + dd	
RETI		Return from interrupt POP SR&PC	

Table 5.2 Instruction List

BWL	LD	reg, reg	BWL	INC	imm3, reg	---	NOP	
BWL	LD	reg, imm		DEC	imm3, mem.B/W			
BWL	LD	reg, mem						
BWL	LD	mem, reg				---	EI	[imm3]
BW-	LD	mem, imm				---	DI	
BW-	LD	(nn), mem	BW-	MUL	reg, reg	-W-	*PUSH	SR
BW-	LD	mem, (nn)		*MULS	reg, imm	-W-	*POP	SR
				DIV	reg, mem	---	SWI	[imm3]
				*DIVS		---	HALT	
BWL	PUSH	reg/F				BWL	*LDC	CTRL – R, reg
BW-	PUSH	imm	-W-	*MULA	reg	BWL	*LDC	reg, CTRL – R
BW-	PUSH	mem				B--	*LDX	(n), n
			-W-	*MINC1	imm, reg			
BWL	POP	reg/F	-W-	*MINC2	imm, reg	--L	*LINK	reg, dd
BW-	POP	mem	-W-	*MINC4	imm, reg	--L	*UNLK	reg
			-W-	*MDEC1	imm, reg	---	*LDF	imm3
			-W-	*MDEC2	imm, reg	---	*INCF	
-WL	LDA	reg, mem	-W-	*MDEC4	imm, reg	---	*DECF	
-WL	LDAR	reg, PC + dd				BW-	*SCC	cc, reg
			BW-	NEG	reg			
			BW-	CPL	reg	BWL	RLC	imm, reg
			-WL	*EXTZ	reg		RRC	A, reg
B--	EX	F, F’	-WL	*EXTS	reg		RL	mem. B/W
BW-	EX	reg, reg	B--	DAA	reg		RR	
BW-	EX	mem, reg	-WL	*PAA	reg		SLA	
							SRA	
							SLL	
-W-	*MIRR	reg	BW-	*LDCF	imm, reg		SRL	
				*STCF	A, reg			
				*ANDCF	imm, mem.B	B--	RLD	[A,] mem
				*ORCF	A, mem.B	B--	RRD	[A,] mem
BW-	LDI			*XORCF				
BW-	LDIR							
BW-	LDD		---	RCF		---	JR	[cc,] PC + d
BW-	LDDR		---	SCF		---	JRL	[cc,] PC + dd
			---	CCF		---	JP	[cc,] mem
			---	*ZCF		---	CALR	PC + dd
BW-	CPI					---	CALL	[cc,] mem
BW-	CPIR		BW-	BIT	imm, reg			
BW-	CPD			RES	imm, mem.B	BW-	DJNZ	[reg], PC + d
BW-	CPDR			SET				
				*CHG		---	RET	[cc]
				TSET		---	*RETD	dd
BWL	ADD	reg, reg				---	RETI	
	ADC	reg, imm	-W-	*BS1F	A, reg			
	SUB	reg, mem		*BS1B				
	SBC	mem, reg						
	CP	mem, imm.B/W						
	AND							
	OR							
	XOR							

B = Byte (8 bits), W = Word (16 bits), L = Long-Word (32 bits).

* : Indicates instruction added to the TLCS-90 series.

[] : Indicates can be omitted.

6. Data Formats

The TLCS-900 series can handle 1/4/8/16/32-bit data.

(1) Register Data Format

<Data image>

1 bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCD	Upper				Lower				Upper				Lower				Upper				Lower				Upper				Lower			
Byte	MSB							LSB	MSB							LSB	MSB							LSB	MSB							LSB
Word	MSB															LSB	MSB															LSB
Long word	MSB																															LSB

Note 1 : To access the parts indicated by ▨, the instruction code is one byte longer than when accessing the other parts.

(2) Memory Data Format

<Data image>

1 bit	Address n	7	6	5	4	3	2	1	0

Byte	Address n	MSB LSB
Word	Address n	Lower LSB
	n + 1	MSB Upper
Long word	Address n	Lower LSB
	n + 1	
	n + 2	
	n + 3	MSB Upper

Note 2 : There are no restrictions on the location of word or long word data in memory. They can be located from even or odd numbered address.

Note 3 : When the PUSH instruction is used to save data to the stack area, the stack pointer is decremented, then the data is saved.

Example: PUSH HL; XSP←XSP − 2
(XSP) ←L
(XSP + 1) ←H

This is the same in register indirect pre-decrement mode. The order is reversed in the TLCS-90 series: data is saved first, then the stack pointer is decremented.

Example: PUSH HL; (XSP − 1) ←H
(XSP − 2) ←L
XSP←XSP − 2

(3) Dynamic Bus Sizing

The TLCS-900 series can switch between 8- and 16-bit data buses dynamically during each bus cycle. This is called dynamic bus sizing. The function enables external memory extension using both 8- and 16-bit data bus memories. Products with a built-in chip select/wait controller can control external data bus size for each address area.

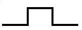
Table 6.1 Dynamic Bus Sizing

Operand data size	Operand start address	Data size at memory side	CPU address	CPU data	
				D15 to D8	D7 to D0
8 bits	2n + 0 (even)	8 bits	2n + 0	xxxxx	b7 to b0
		16 bits	2n + 0	xxxxx	b7 to b0
	2n + 1 (odd)	8 bits	2n + 1	xxxxx	b7 to b0
		16 bits	2n + 1	b7 to b0	xxxxx
16 bits	2n + 0 (even)	8 bits	2n + 0	xxxxx	b7 to b0
			2n + 1	xxxxx	b15 to b8
		16 bits	2n + 0	b15 to b8	b7 to b0
	2n + 1 (odd)	8 bits	2n + 1	xxxxx	b7 to b0
			2n + 2	xxxxx	b15 to b8
		16 bits	2n + 1	b7 to b0	xxxxx
			2n + 2	xxxxx	b15 to b8
32 bits	2n + 0 (even)	8 bits	2n + 0	xxxxx	b7 to b0
			2n + 1	xxxxx	b15 to b8
			2n + 2	xxxxx	b23 to b16
			2n + 3	xxxxx	b31 to b24
		16 bits	2n + 0	b15 to b8	b7 to b0
			2n + 2	b31 to b24	b23 to b16
	2n + 1 (odd)	8 bits	2n + 1	xxxxx	b7 to b0
			2n + 2	xxxxx	b15 to b8
			2n + 3	xxxxx	b23 to b16
			2n + 4	xxxxx	b31 to b24
		16 bits	2n + 1	b7 to b0	xxxxx
			2n + 2	b23 to b16	b15 to b8
			2n + 4	xxxxx	b31 to b24

xxxxx : During read, indicates the data input to the bus are ignored. During write, indicates the bus is at high impedance and the write strobe signal is non-active.

(4) Internal Data Bus Format

With the TLCS-900 series, the CPU and the internal memory (built-in ROM or RAM) are connected via a 16-bit internal data bus. The internal memory operates with 0 wait. The CPU and the built-in I/Os are connected using an 8-bit internal data bus. This is because the built-in I/O access speed has little influence on the overall system operation speed.

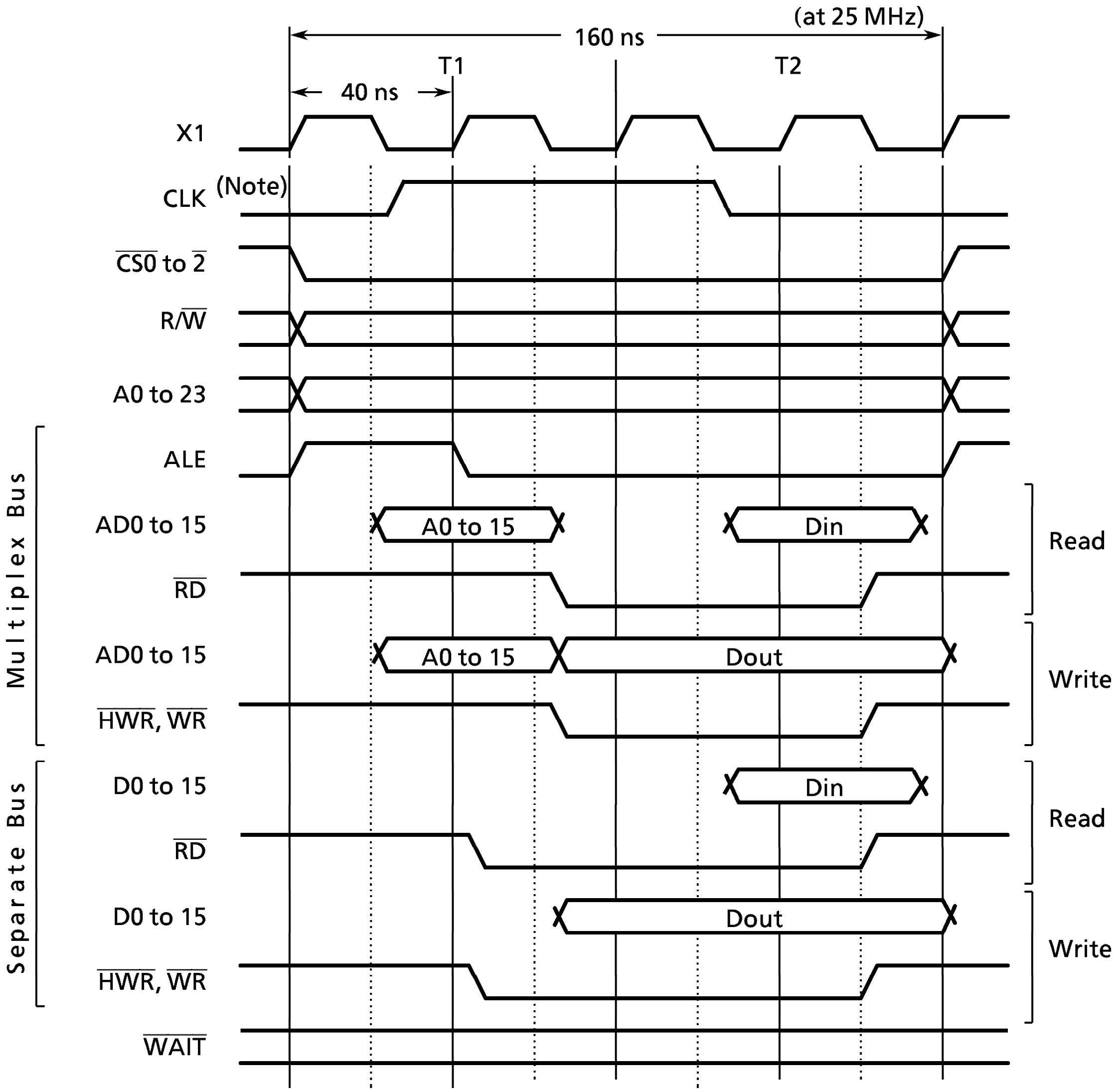
Overall system operation speed depends largely on the speed of program memory access. The built-in I/O operates in sync with the signal phase of the CLK pin. It is synchronized so that the CLK rises () in the middle of the bus cycle. (Figure 7.1 shows signal phases.) If the CLK is “1” when the ALE signal rises, 1 wait is inserted automatically for synchronization.

7. Basic Timings

The TLCS-900 series runs the following basic timings.

- Read cycle
- Write cycle
- Dummy cycle
- Interrupt receive timing
- Reset

Figures 7.1 to 7.10 show the basic timings.



Note : CLK outputs are not always the same as the above phases.

Figure 7.1 0 WAIT Read/Write Cycle

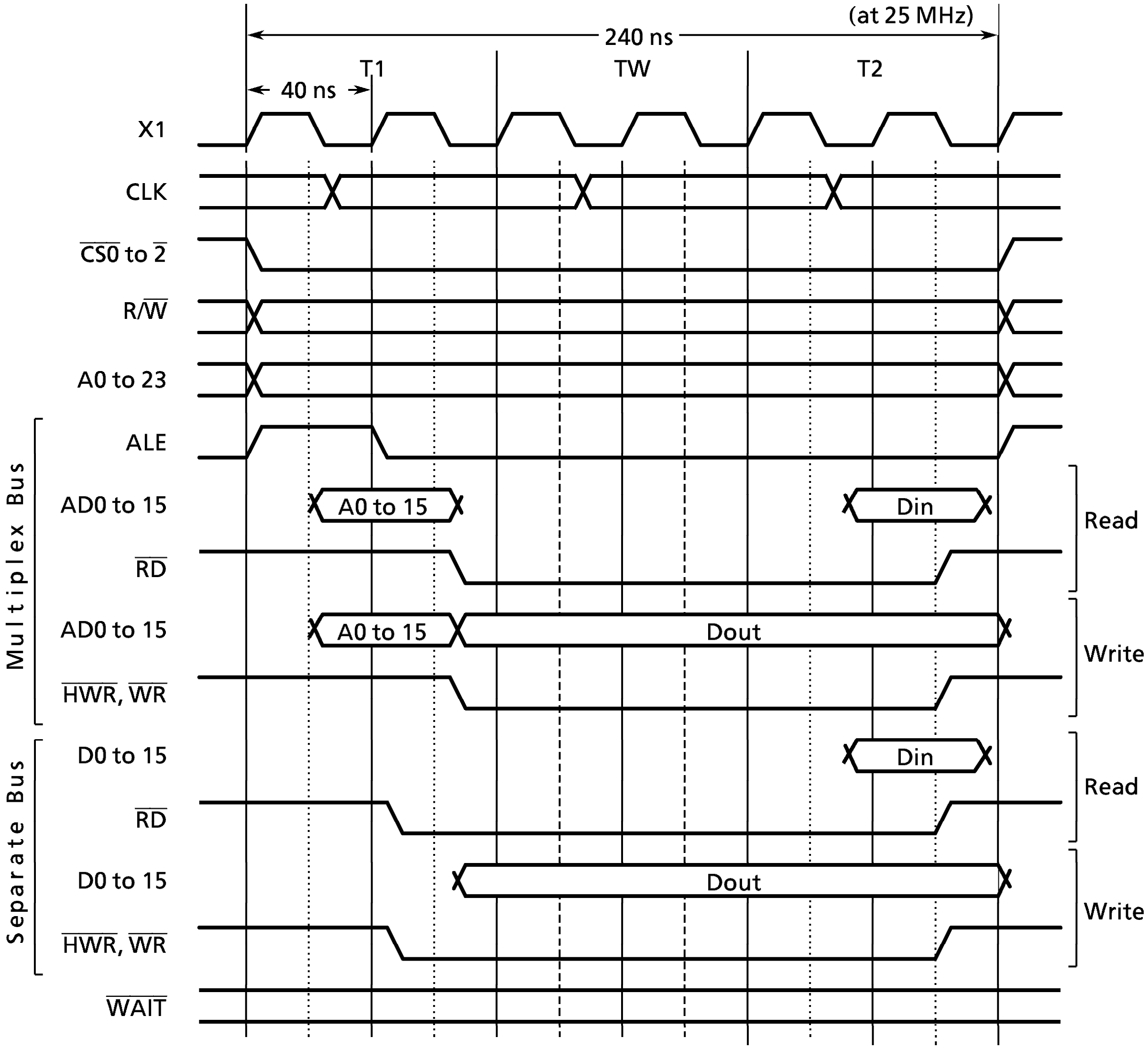


Figure 7.2 1WAIT Read/Write Cycle

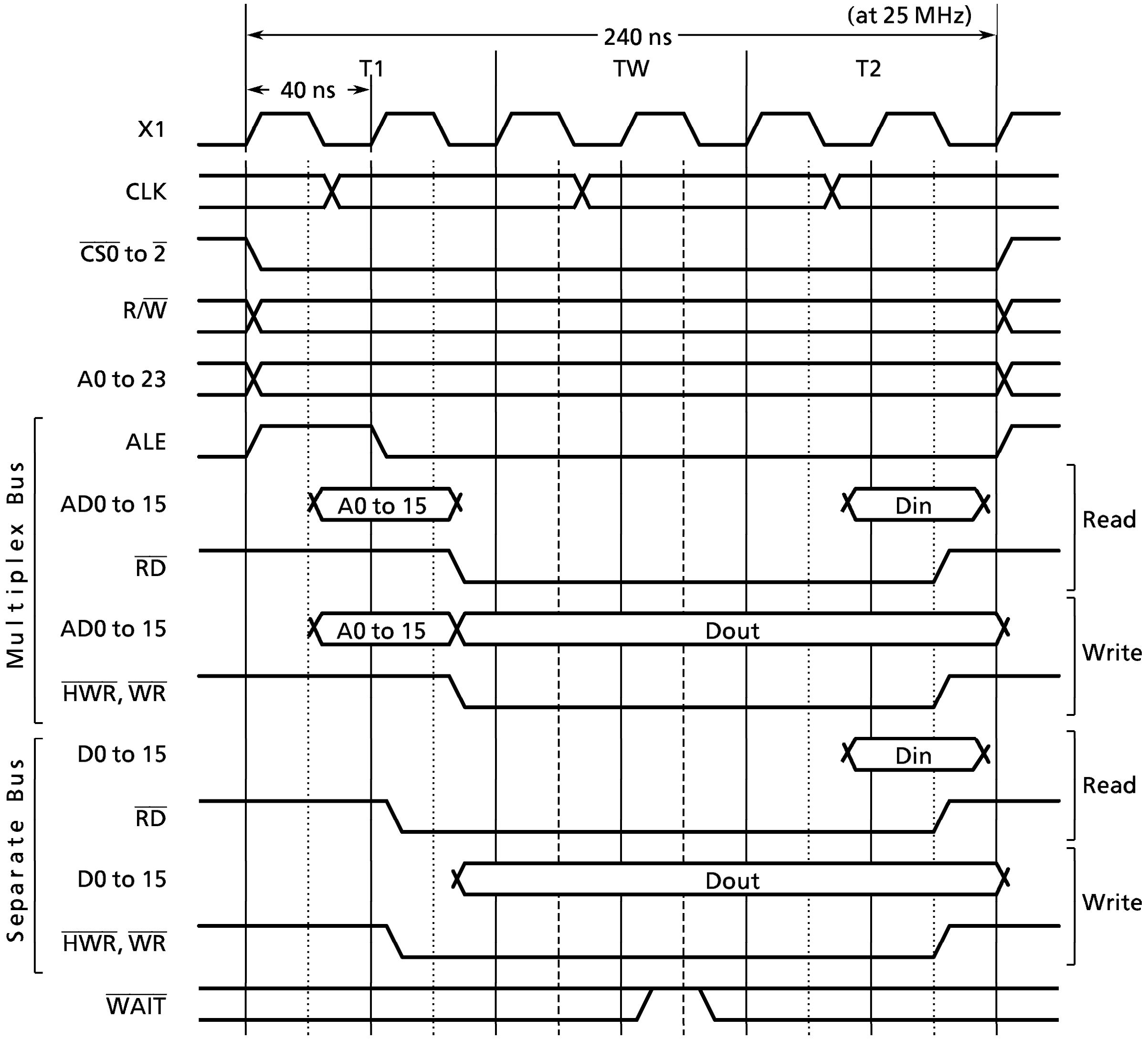


Figure 7.3 1WAIT + n Read/Write Cycle (n = 0)

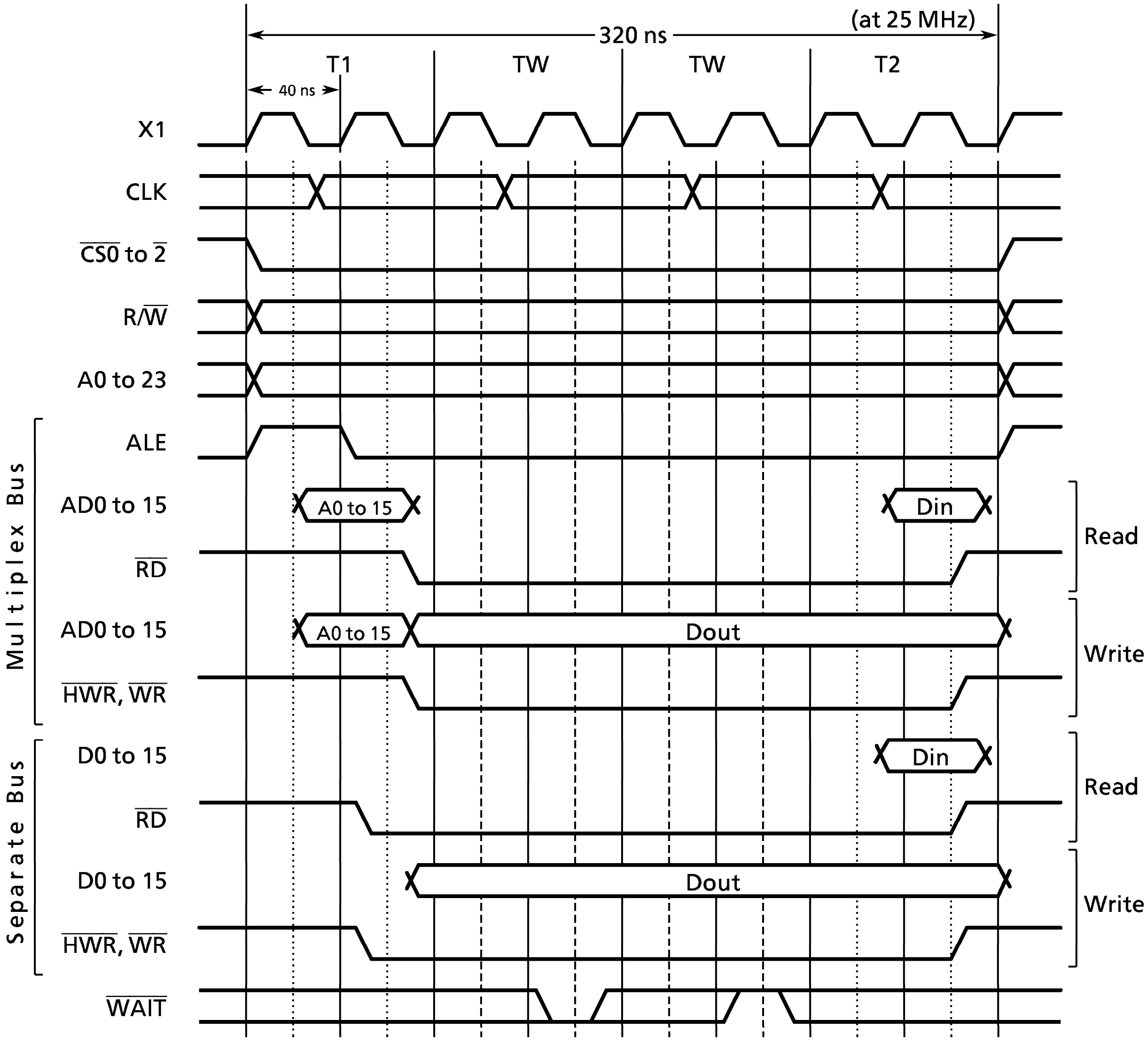


Figure 7.4 1WAIT + n Read/Write Cycle (n = 1)

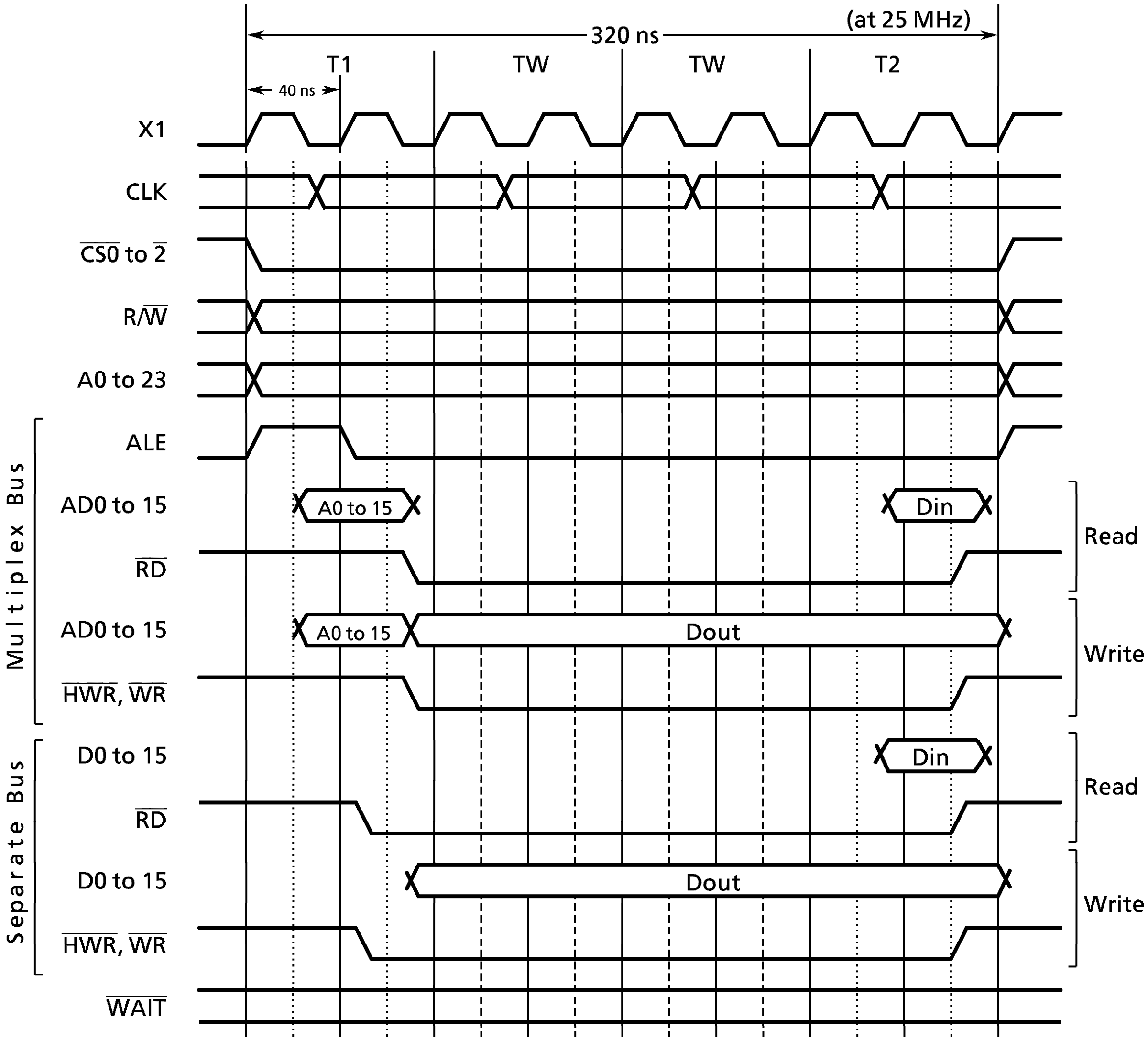


Figure 7.5 2WAIT Read/Write Cycle

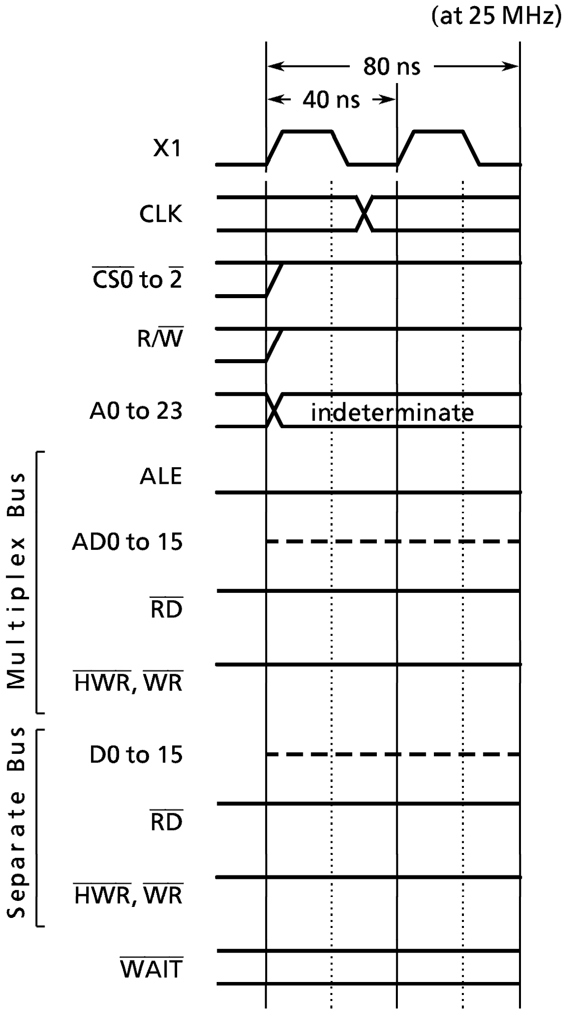
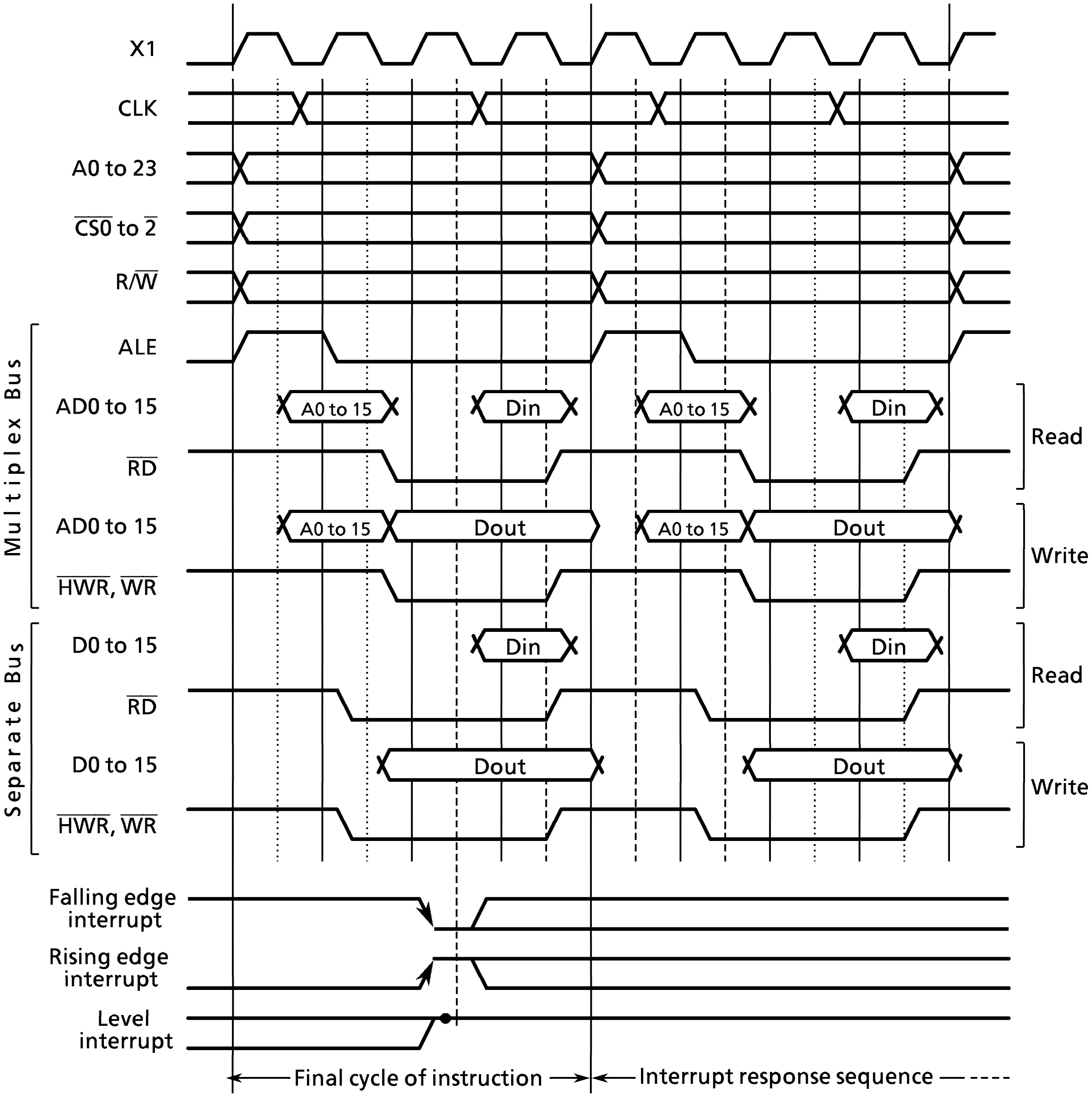


Figure 7.6 1 State Dummy Cycle



Note : This timing chart is a theoretical example. In practice, due to the operation of the bus interface unit in the CPU, external bus and internal interrupt receive timings do not correspond one to one.

Figure 7.7 Interrupt Receive Timing

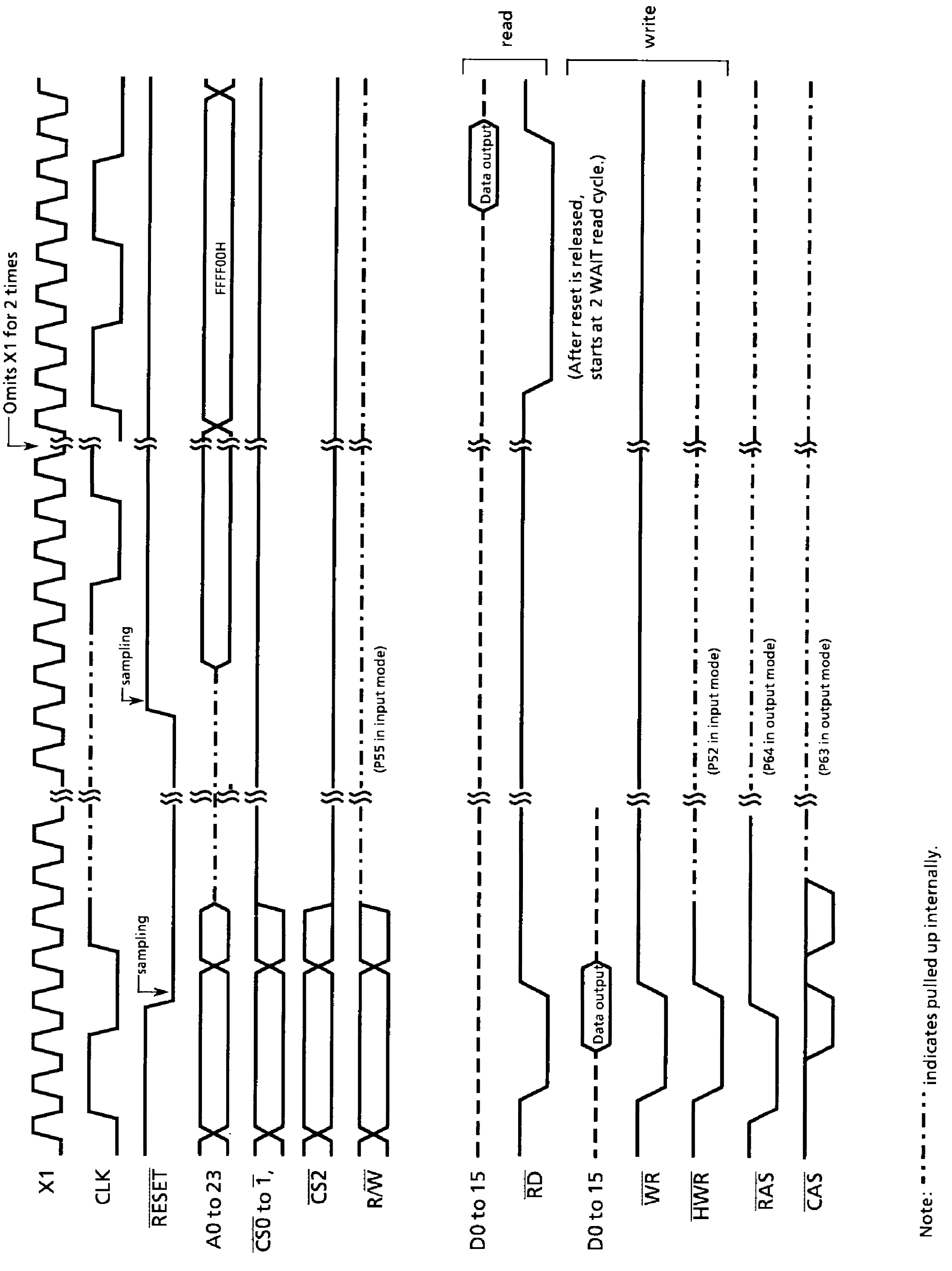


Figure 7.8 Reset Timings (external ROM operation : TMP95C061)