Zilog - Z8038018FSC00TR Datasheet

Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

Applications of Embedded - Microprocessors

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

Details	
Product Status	Obsolete
Core Processor	Z380
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	18MHz
Co-Processors/DSP	-
RAM Controllers	DRAM
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5.0V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	100-QFP
Supplier Device Package	100-QFP
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8038018fsc00tr

Revision History

Each instance in Revision History reflects a change to this document from its previous revision. For more details, refer to the corresponding pages and appropriate links in the table below.

Date	Revision Level	Description	Page No
July 2008	02	Updated format to the latest PS template	All
March 2001	01	Original Issue	All

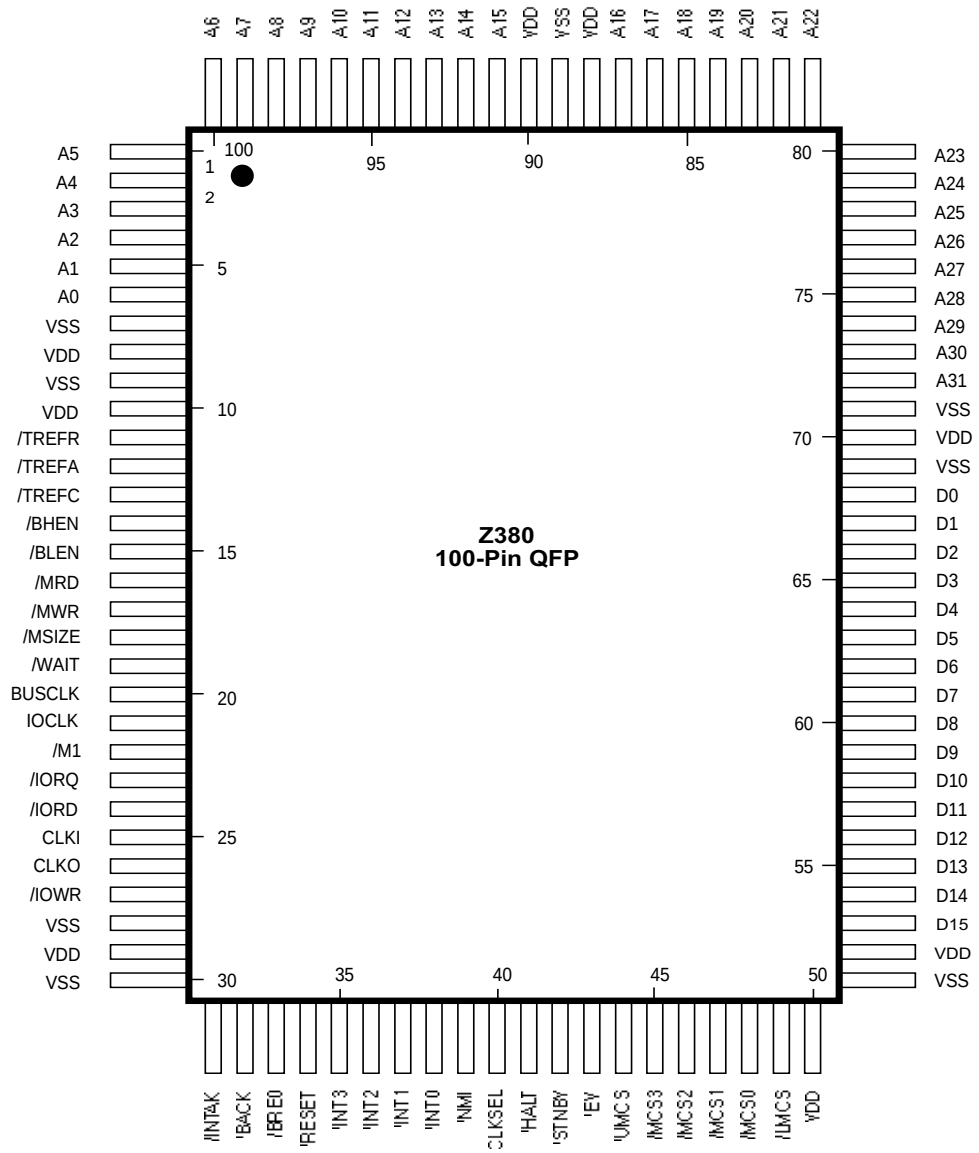

Figure 2. 100-Pin QFP Pin Assignments

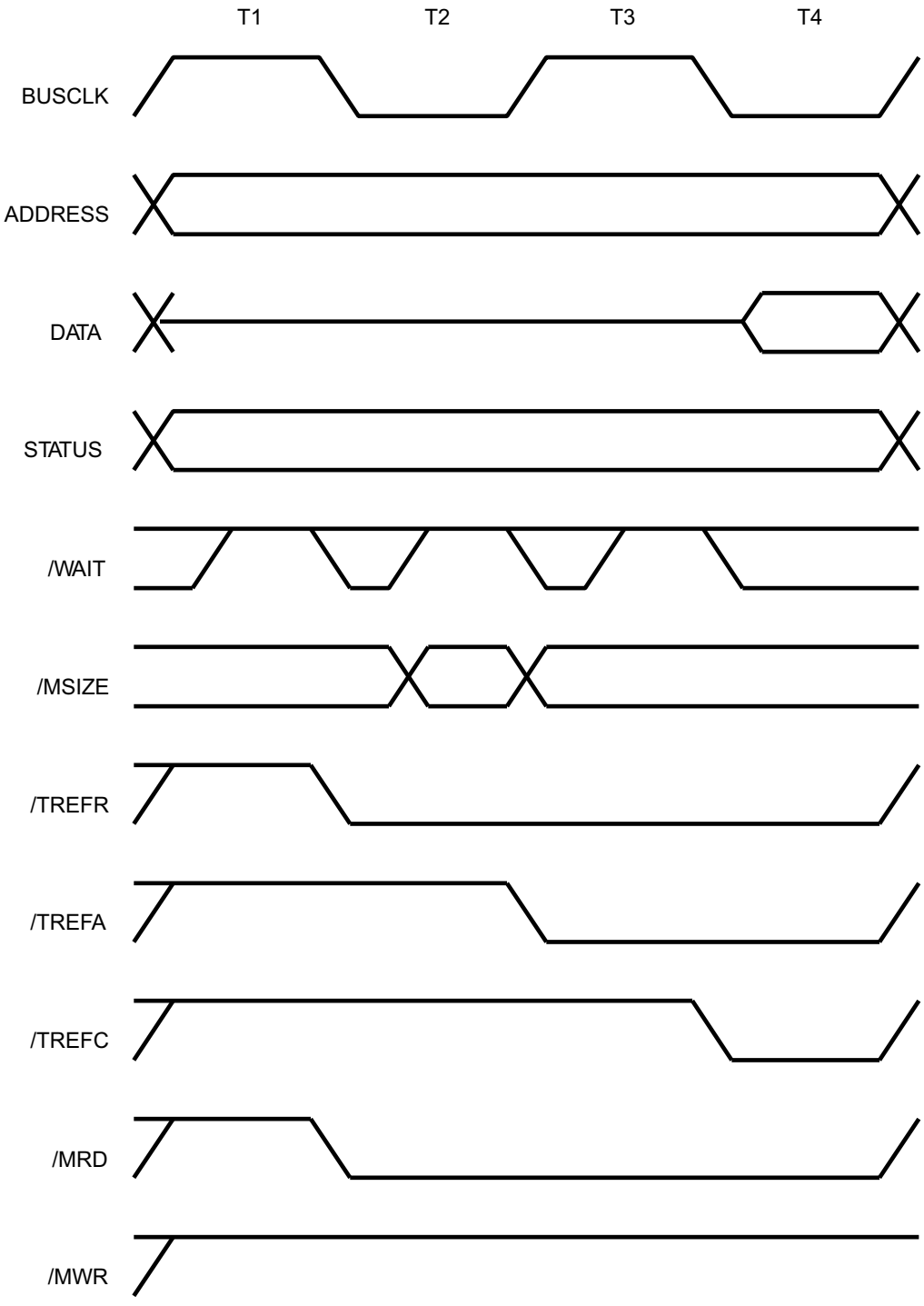

Figure 3. Read Cycle, No Waits

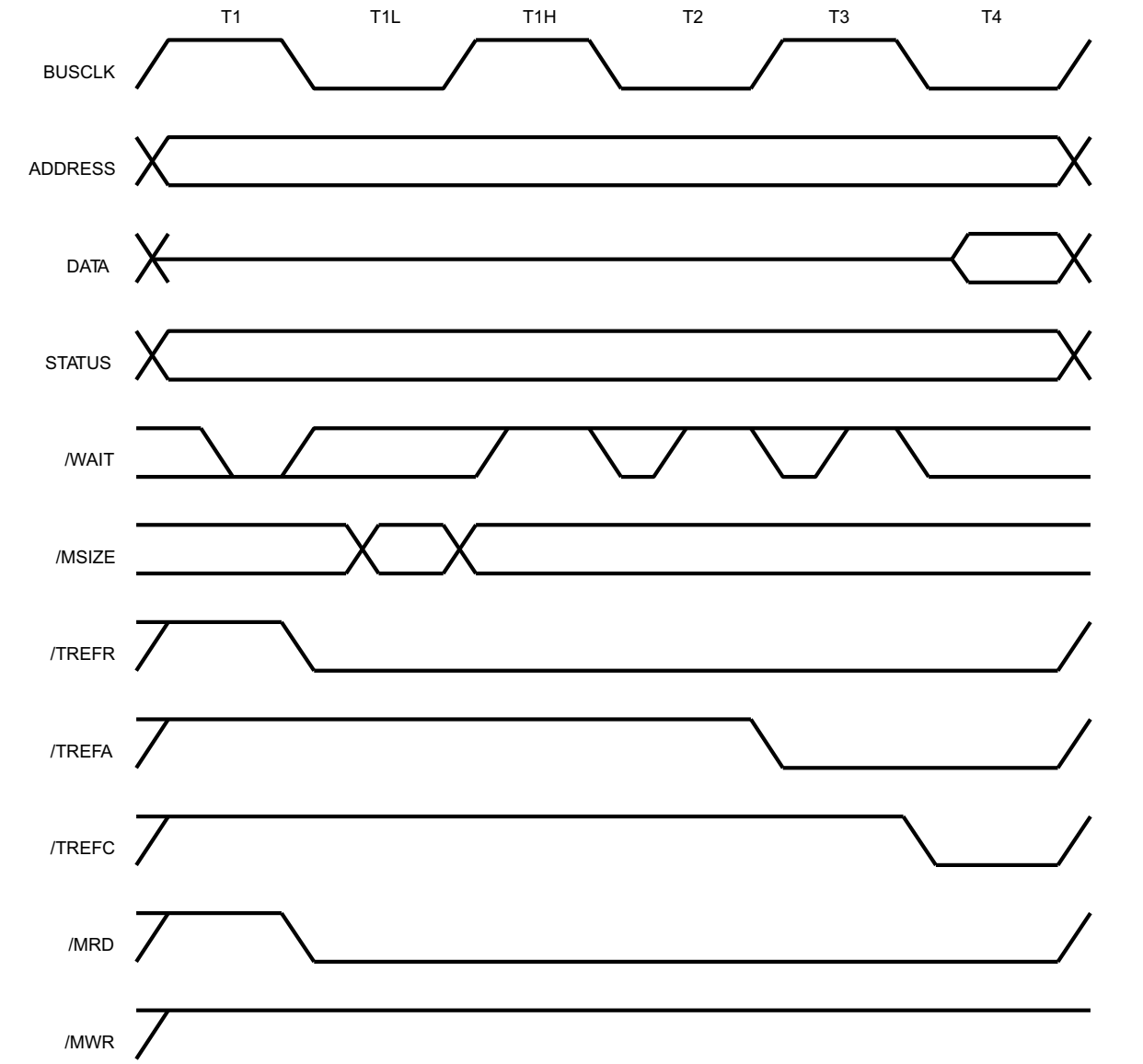

Figure 4. Read Cycle, T1 Wait

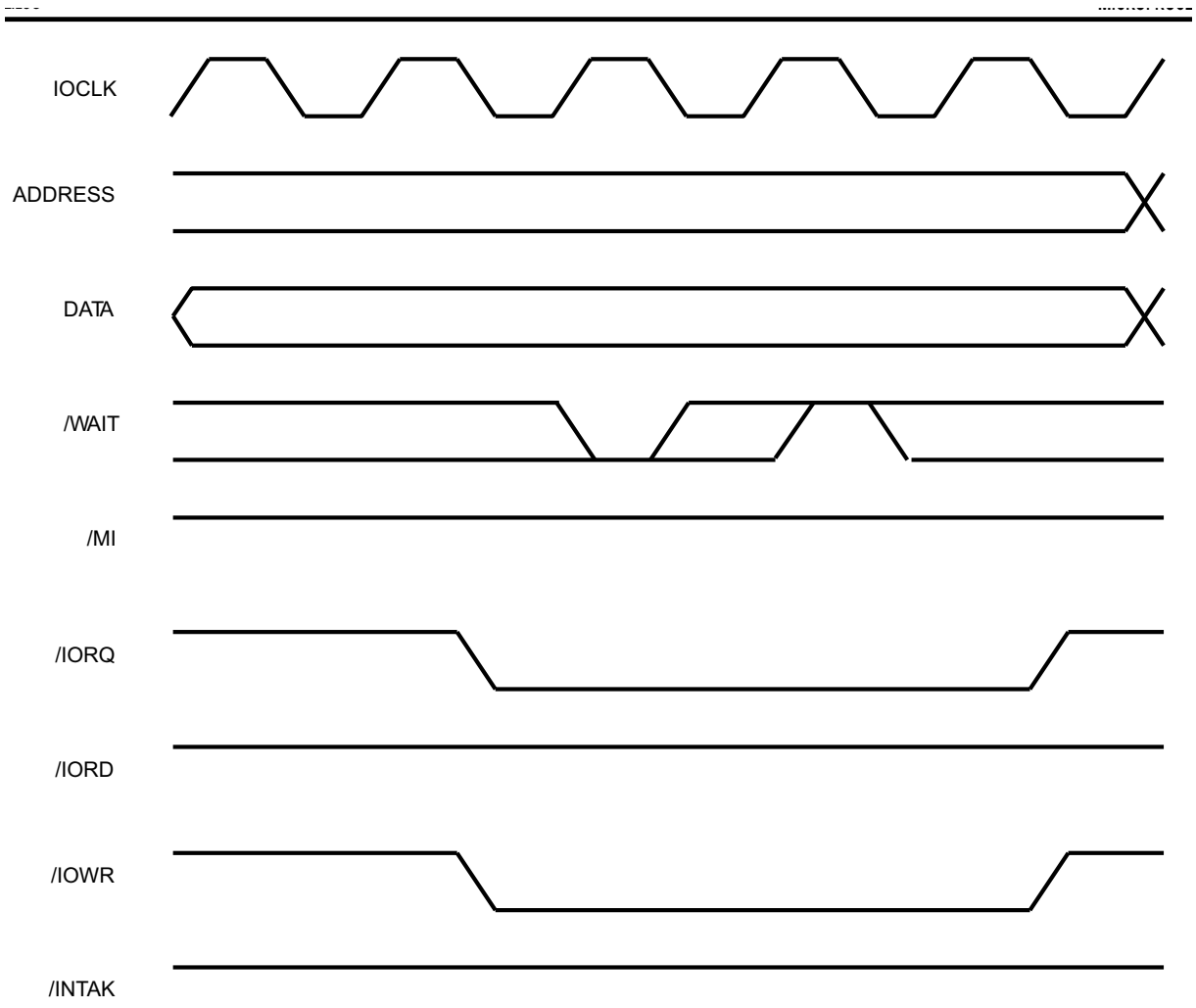

Figure 20. I/O Write Cycle, T1 Wait

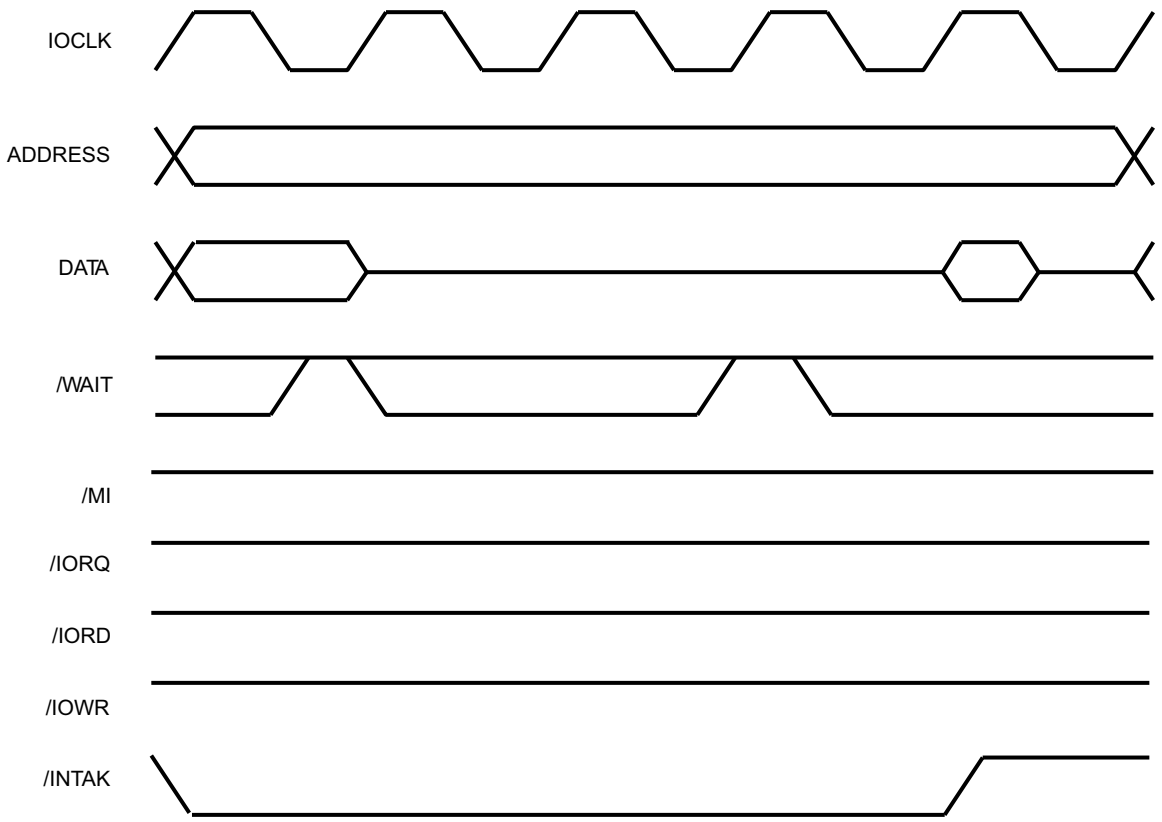

Figure 22. Interrupt Acknowledge Cycle, /INT3-1

An interrupt acknowledge transaction for /INT0 is five IOCLK cycles long unless extended by Wait states. /WAIT is sampled at two separate points during the transaction. /WAIT is first sampled at the end of the first IOCLK cycle during the transaction. Wait states inserted here allow the external daisy-chain between peripherals with a longer time to settle before the interrupt vector is requested. /WAIT is then sampled at the end of the fourth IOCLK cycle to delay the point at which the interrupt vector is read by the Z380 MPU, after it has been requested.

The interrupt vector may be either eight or sixteen bits, under program control, and is latched by the falling edge of IOCLK in the last cycle of the interrupt acknowledge transaction. When using Mode 0 interrupts, where the Z380 MPU fetches an instruction from the interrupting device, these fetches are always eight bits wide and are transferred over D7-D0.

CPU ARCHITECTURE

The Central Processing Unit (CPU) of the Z380 MPU is a binary-compatible extension of the Z80 CPU and Z180 CPU architectures. High throughput rates for the Z380 CPU are achieved by a high clock rate, high bus bandwidth and instruction fetch/execute overlap. Communicating to the external world through an 8-or 16-bit data bus, the Z380 CPU is a full 32-bit machine internally, with a 32-bit ALU and 32-bit registers.

Modes Of Operation

The Z380 CPU can operate in either Native or Extended mode, as controlled by a bit in the Select Register (SR). In Native mode (the Reset configuration), all address manipulations are performed modulo 65536 (16 bits). In this mode the Program Counter (PC) only increments across 16 bits, all address manipulation instructions (increment, decrement, add, subtract, indexed, stack relative, and PC relative) only operate on 16 bits, and the Stack Pointer (SP) only increments and decrements across 16 bits. The program counter high-order word is left at all zeros, as is the high-order words of the stack pointer and the I register. Thus Native mode is fully compatible with the Z80 CPU's 64 Kbyte address space. It is still possible to address memory outside of the 64 Kbyte address space for data storage and retrieved in Native mode, however, direct addresses, indirect addresses, and the high-order word of the SP, I and the IX and IY registers may be loaded with non-zero values. But executed code and interrupt service routines must reside in the lowest 64 Kbytes of the address space.

In Extended mode, however, all address manipulation instructions operate on 32 bits, allowing access to the entire 4 Gbyte address space of the Z380 MPU. In both Native and Extended modes, the Z380 CPU drives all 32 bits of the address onto the external address bus; only the width of manipulated addresses distinguish Native from Extended mode. The Z380 CPU implements one instruction to allow switching from Native to Extended mode, but once in Extended mode, only Reset returns the Z380 MPU to Native mode. This restriction applies because of the possibility of "misplacing" interrupt service routines or vector tables during the translation from Extended mode back to Native mode.

In addition to Native and Extended mode, which is specific to memory space addressing, the Z380 MPU can operate in either Word or Long Word mode specific to data load and exchange operations. In Word mode (the reset configuration), all word load and exchange operations manipulate 16-bit quantities. For example, only the low-order words of the source and destination are exchanged in an exchange operation, with the high-order words unaffected. In Long Word mode, all 32 bits of the source and destination are directives to allow switching between Word and Long Word mode; SETC LW (Set Control Long Word) and RESC LW (Reset Control Long Word) perform a global switch, while DDIR W, DDIR LW and their variants are decoder directives that select a particular mode only for the instruction that they precede.

Note that all word data arithmetic (as opposed to address manipulation arithmetic), rotate, shift and logical operations are always in 16-bit quantities. They are not controlled by either the Native/Extended or Word/Long Word selections. The exceptions to the 16-bit

quantities are, of course, those multiply and divide operations with 32-bit products or dividends.

Lastly, all word Input/Output operations are performed on 16-bit values.

Address Spaces

The Z380 CPU architecture supports five distinct address spaces corresponding to the different types of locations that can be accessed by the CPU. These five address spaces are: CPU register space, CPU control register space, memory address space, and I/O address space (on-chip and external).

CPU Register Space

The CPU register space is shown in Figure 33 and consists of all of the registers in the CPU register file. These CPU registers are used for data and address manipulation, and are an extension of the Z80 CPU register set, with four sets of this extended Z80 CPU register set present in the Z380 CPU. Access to these registers is specified in the instruction, with the active register set selected by bits in the Select Register (SR) in the CPU control register space.

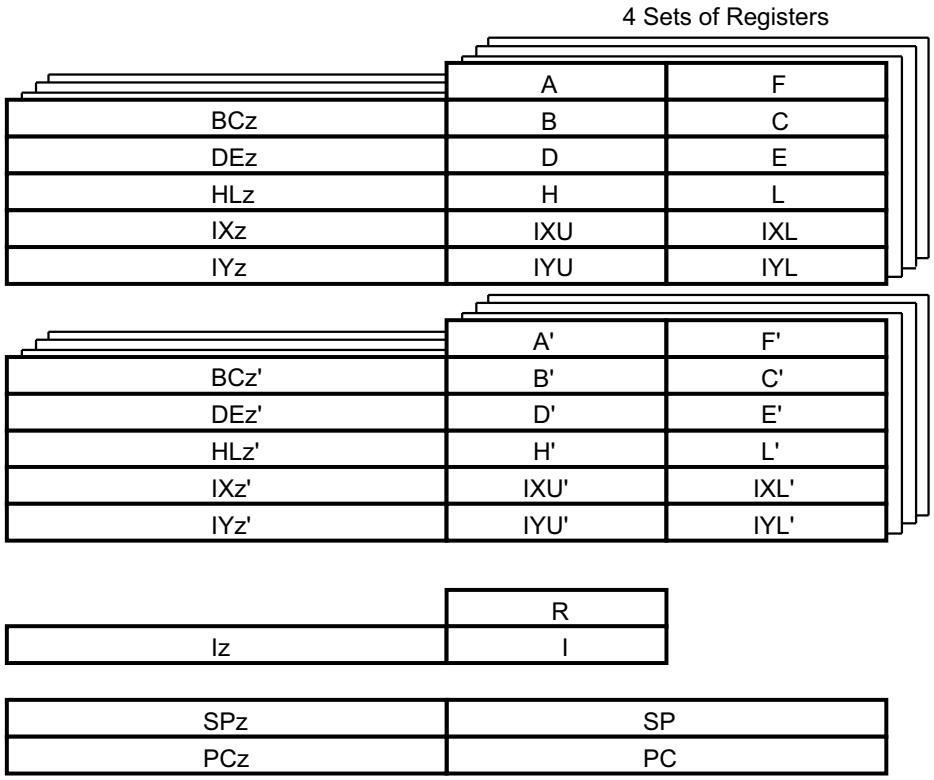

Figure 33. Register Set

Carry (C). This flag is set when an add instruction generates a carry or a subtract instruction generates a borrow. Certain logical, rotate and shift instructions affect the Carry flag.

Add/Subtract (N). This flag is used by the Decimal Adjust Accumulator instruction to distinguish between add and subtract operations. The flag is set for subtract operations and cleared for add operations.

Parity/Overflow (P/V). During arithmetic operations this flag is set to indicate a two's complement overflow. During logical and rotate operations, this flag is set to indicate even parity of the result or cleared to indicate odd parity.

Half Carry (H). This flag is set if an 8-bit arithmetic operation generates a carry or borrow between bits 3 and 4, or if a 16-bit operation generates a carry or borrow between bits 11 and 12, or if a 32-bit operation generates a carry or borrow between bits 27 and 28. This bit is used to correct the result of a packed BCD addition or subtract operation.

Zero (Z). This flag is set if the result of an arithmetic or logical operation is a zero.

Sign (S). This flag stores the state of the most significant bit of the accumulator.

Index Registers

The four index registers, IX, IX', IY and IY', each hold a 32-bit base address that is used in the Indexed addressing mode. The Index registers can also function as general-purpose registers with the upper and lower byte of the lower 16 bits being accessed individually. These byte registers are called IXU, IXU', IXL and IXL' for the IX and IX' registers, and IYU, IYU', IYL and IYL' for the IY and IY' registers.

Interrupt Register

The Interrupt register (I) is used in interrupt modes 2 and 3 for /INT0 to generate a 32-bit indirect address to an interrupt service routine. The I register supplies the upper twenty-four or sixteen bits of the indirect address and the interrupting peripheral supplies the lower eight or sixteen bits. In the Assigned Vectors mode for /INT1-3 the upper sixteen bits of the vector are supplied by the I register; bits 15-9 are the assigned vector base and bits 8-0 are the assigned vector unique to each of /INT1-3.

Program Counter

The Program Counter (PC) is used to sequence through instructions in the currently executing program and to generate relative addresses. The PC contains the 32-bit address of the current instruction being fetched from memory. In the Native mode, the PC is effectively only 16 bits long, as carries from bit 15 to bit 16 are inhibited in this mode. In Extended mode, the PC is allowed to increment across all 32 bits.

R Register

The R register can be used as a general-purpose 8-bit read/write register. The R register is not associated with the refresh controller and its contents are changed only by the user.

Mnemonic	Symbolic Operation	Flags S	Z	x	H	x	P/V	N	C	Opcode 76 543 210	HEX	# of Bytes	Execute Time	Notes
LD dd,nn	dd ← nn	•	•	x	•	x	•	•	•	00 dd0 001 ⟵ n ⟶ ⟵ n ⟶		3	2	L1,I
LD XY,nn	XY ← nn	•	•	x	•	x	•	•	•	11 y11 101 00 100 001 ⟵ n ⟶ ⟵ n ⟶	21	4	2	L1,I
LD HL,(nn)	H ← (nn+1) L ← (nn)	•	•	x	•	x	•	•	•	00 101 010 ⟵ n ⟶ ⟵ n ⟶	2A	3	3+r	L1,I
LD dd,(nn)	ddh ← (nn+1) ddl ← (nn)	•	•	x	•	x	•	•	•	11 101 101 01 dd1 011 ⟵ n ⟶ ⟵ n ⟶	ED	4	3+r	L1,I
LD XY,(nn)	XYU ← (nn+1) XYL ← (nn)	•	•	x	•	x	•	•	•	11 y11 101 00 101 010 ⟵ n ⟶ ⟵ n ⟶	2A	4	3+r	L1,I
LD (nn),HL	(nn+1) ← H (nn) ← L	•	•	x	•	x	•	•	•	00 100 010 ⟵ n ⟶ ⟵ n ⟶	22	3	4+w	L1,I
LD (nn),dd	(nn+1) ← ddh (nn) ← ddl	•	•	x	•	x	•	•	•	11 101 101 01 dd0 011 ⟵ n ⟶ ⟵ n ⟶	ED	4	4+w	L1,I
LD (nn),XY	(nn+1) ← XYU (nn) ← XYL	•	•	x	•	x	•	•	•	11 y11 101 00 100 010 ⟵ n ⟶ ⟵ n ⟶	22	4	4+w	L1,I
LD W(pp),nn	(pp+1) ← nh (pp) ← nl	•	•	x	•	x	•	•	•	11 101 101 00 pp0 110 ⟵ n ⟶ ⟵ n ⟶	ED	4	3+w	L1,I
LD pp,(uu)	pph ← (uu+1) ppl ← (uu)	•	•	x	•	x	•	•	•	11 011 101 00 pp1 1uu	DD	2	2+r	L1
LD (pp),uu	(pp+1) ← uuh (pp) ← uul	•	•	x	•	x	•	•	•	11 111 101 00 pp1 1uu	FD	2	3+w	L1
LD SP,HL	SP ← HL	•	•	x	•	x	•	•	•	11 111 001	F9	1	2	L1
LD SP,XY	SP ← XY	•	•	x	•	x	•	•	•	11 y11 101 11 111 001	F9	2	2	L1
LD pp,UU	pp ← UU	•	•	x	•	x	•	•	•	11 UU1 101 00 pp0 010		2	2	L1
LD XY,pp	XY ← pp	•	•	x	•	x	•	•	•	11 y11 101 00 pp0 111		2	2	L1
LD IX,IY	IX ← IY	•	•	x	•	x	•	•	•	11 011 101 00 100 111	DD 27	2	2	L1

Mnemonic	Symbolic Operation	Flags S	Z	x	H	x	P/V	N	C	Opcode 76	543	210	HEX	# of Bytes	Execute Time	Notes
LD (SP+d),pp	(SP+d)h ← pph (SP+d)l ← ppl	•	•	x	•	x	•	•	•	11 11 ←— d —→ 00	011 001 pp1	101 011 001	DD CB 	4	5+w	L1, I
LD (SP+d),XY	(SP+d)h ← XYU (SP+d)l ← XYL	•	•	x	•	x	•	•	•	11 11 ←— d —→ 00	y11 001 101	101 011 001	 CB 29	4	5+w	L1, I
LD [W] I,HL	I ← HL	•	•	x	•	x	•	•	•	11 01	011 000	101 111	DD 47	2	2	L1
LD [W] HL,I	HL ← I	•	•	x	•	x	•	•	•	11 01	011 010	101 111	DD 57	2	2	L1

dd	Pair
00	BC
01	DE
10	HL
11	SP

qq	Pair
00	BC
01	DE
10	HL
11	AF

pp,uu	Pair
00	BC
01	DE
11	HL

y	XY
0	IX
1	IY

Notes:
Instructions in ***Italic*** face are Z380 new instructions, instructions with **underline** are Z180 original instructions.
I: This instruction may be used with DDIR Immediate instructions.
L1: In Long Word mode, this instruction loads in 32 bits; dst(31-0) ← src(31-0)

8-BIT BIT SET, RESET, AND TEST GROUP

Mnemonic	Symbolic Operation	Flags S	Z	x	H	x	P/V	N	C	Opcode 76	543	210	HEX	# of Bytes	Execute Time	Notes
BIT b,r	Z ← rb	•	↕	x	1	x	•	0	•	11	001	011	CB	2		
										01	b	r				
BIT b,(HL)	Z ← (HL)b	•	↕	x	1	x	•	0	•	11	001	011	CB	2		
										01	b	110				
BIT b,(XY+d)	Z ← (XY+d)b	•	↕	x	1	x	•	0	•	11	y11	101		4		I
										11	001	011	CB			
										⟵	d	⟶				
										01	b	110				
SET b,r	rb ← 1	•	•	x	•	x	•	•	•	11	001	011	CB	2		
										(11)	b	r				
SET b,(HL)	(HL)b ← 1	•	•	x	•	x	•	•	•	11	001	011	CB	2		
										(11)	b	110				
SET b,(XY+d)	(XY+d)b ← 1	•	•	x	•	x	•	•	•	11	y11	101		4		I
										11	001	011	CB			
										⟵	d	⟶				
										(11)	b	110				
RES b,m	mb ← 0									(10)						

To form new opcode replace (11) of SET b,s with (10). s is any of r,(HL), (XY+d).
The notation mb indicates location m, bit b(0~7)

r	Reg
000	B
001	C
010	D
011	E
100	H
101	L
111	A

y	XY
0	IX
1	IY

Notes:
Instructions in ***Italic*** face are Z380 new instructions, instructions with **underline** are Z180 original instructions.
I: This instruction may be operate with DDIR Immediate instructions.

Mnemonic	Symbolic Operation	Flags S	Z	x	H	x	P/V	N	C	Opcode 76	543	210	HEX	# of Bytes	Execute Time	Notes
RETN	Return from NMI	•	•	x	•	x	•	•	•	11 01	101 000	101 101	ED 45	2	2+r	N,X4,(10)
RST p	(SP-1) ← PCh (SP-2) ← PCl SP ← SP-2 PCh ← 0 PCl ← p	•	•	x	•	x	•	•	•	11	t	111		1	4+w	N,X3,X5

cc	Condition
000	NZ (Non-zero)
001	Z (Zero)
010	NC (Non-carry)
011	C (Carry)
100	PO (Parity Odd), or NV (Non-Overflow)
101	PE (Parity Even), or V (Overflow)
110	P (Sign positive), or NS (No sign)
111	M (Sign negative), or S (Sign)

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

Notes:
Instructions in ***Italic*** face are Z380 new instructions, instructions with **underline** are Z180 original instructions.

I: This instruction may be used with DDIR Immediate instructions.
N: In Native mode, this instruction uses addresses modulo 65536.
X3: In Extended mode, this instruction pushes PC(31-16) into the stack before pushing PC(15-0) into the stack.
X4: In Extended mode, this instruction pops PC(31-16) from the stack after poping PC(15-0) from the stack.
X5: In Extended mode, this instruction loads 00h into PC(31-16).
(2) In Extended mode, all return instructions pops PCz from the stack after poping PC from the stack.
(8): ee is a signed two's complement number in the range [-32765, 32770], ee-4 in the opcode provides an effective address of pc+e as PC is incremented by 4 prior to the addition of e.
(9): eee is a signed two's complement number in the range [-8388604, 8388611], eee-5 in the opcode provides an effective address of pc+e as PC is incremented by 5 prior to the addition of e.
(10) RETN loads IFF2 to IFF1.
(11): e is a signed two's complement number in the range [-127, 128], e-3 in the opcode provides an effective address of pc+e as PC is incremented by 3 prior to the addition of e.

16-BIT INPUT AND OUTPUT GROUP

Mnemonic	Symbolic Operation	Flags S	Z	x	H	x	P/V	N	C	Opcode 76	543	210	HEX	# of Bytes	Execute Time	Notes
INW pp,(C)	pp ← (C)	↕	↕	x	0	x	P	0	•	11	011	101	DD	2		
										01	ppp	000				
INAW HL,(nn)	HL(15-0) ← (nn)	•	•	x	•	x	•	•	•	11	111	101	FD	4	3+i	I
										11	011	011	DB			
										←— n —→						
										←— n —→						
INIW	(HL) ← (DE) BC(15-0) ← BC(15-0) - 1 HL ← HL+2	•	↕ (1)	x	•	x	•	1	•	11	101	101	ED	2	2+i+w	N
										11	100	010	E2			
INIRW	(HL) ← (DE) BC(15-0) ← BC(15-0) - 1 HL ← HL+2 Repeat until BC = 0	•	1 (2)	x	•	x	•	1	•	11	101	101	ED	2	(2+i+w)n	N
										11	110	010	F2			
INDW	(HL) ← (DE) BC(15-0) ← BC(15-0) - 1 HL ← HL - 2	•	↕ (1)	x	•	x	•	1	•	11	101	101	ED	2	2+i+w	N
										11	101	010	EA			
INDRW	(HL) ← (DE) BC(15-0) ← BC(15-0) - 1 HL ← HL - 2 Repeat until BC = 0	•	1 (2)	x	•	x	•	1	•	11	101	101	ED	2	(2+i+w)n	N
										11	111	010	FA			
OUTW (C),pp	(C) ← pp	•	•	x	•	x	•	•	•	11	011	101	DD	2	2+o	
										01	ppp	001				
OUTW (C),nn	(C) ← nn	•	•	x	•	x	•	•	•	11	111	101	FD	4	2+o	
										01	111	001	79			
										←— n —→						
										←— n —→						
OUTAW (nn),HL	(nn) ← HL(15-0)	•	•	x	•	x	•	•	•	11	111	101	FD	4	2+o	I
										11	010	011	D3			
										←— n —→						
										←— n —→						
OUTIW	(DE) ← (HL) BC(15-0) ← BC(15-0) - 1 HL ← HL + 2	•	↕ (1)	x	•	x	•	1	•	11	101	101	ED	2	2+o	N
										11	100	011	E3			
OTIRW	BC(15-0) ← BC(15-0) - 1 (DE) ← (HL) HL ← HL + 2 Repeat until B = 0	•	1 (2)	x	•	x	•	1	•	11	101	101	ED	2	2+o	N
										11	110	011	F3			

Lower Memory Chip Select Control

This memory area has its lower boundary at address 000000000H. A user can define the size to be an integer power of two, starting at 4 Kbytes. For example, the lower memory area can be either 4 Kbytes, 8 Kbytes, 16 Kbytes, etc., starting from address 0. The /LMCS signal can be enabled to go active during refresh transactions.

Lower Memory Chip Select Register 0

MA15-MA12 *(Match Address Bits 15-12).* If a match address bit is at logic 1, the corresponding address signal of a memory transaction is compared for a logic 0, as a condition for /LMCS to become active. If the match address bit is at logic 0, the corresponding address signal is not compared (don't care). For example, MA12 determines if A12 should be tested for a logic 0 in memory transactions.

Reserved bits 3-1. Read as 0s, should write to as 0s.

ERF *(Enable for Refresh transactions).* If this bit is programmed to a logic one, /LMCS goes active during refresh transactions.

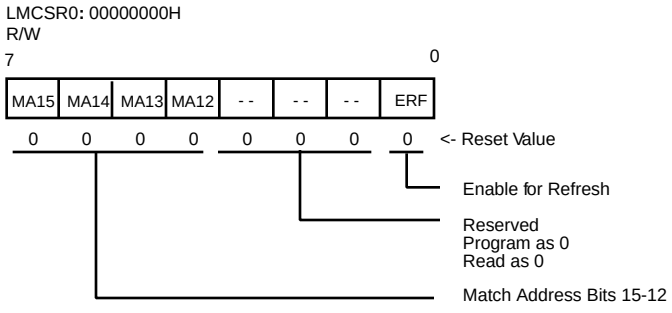

Figure 32. Lower Memory Chip Select Register 0

Lower Memory Chip Select Register 1

MA23-MA16 *(Match Address Bits 23-16).* If a match address bit is at logic 1, the corresponding address signal of a memory transaction is compared for a logic 0, as a condition for /LMCS to become active. If the match address bit is at logic 0, the corresponding address signal is not compared (don't care). For example, MA23 determines if A23 should be tested for a logic 0 in memory transactions. Note that in order for /LMCS to go active in a memory transaction, the /LMCS function has to be enabled in the Memory Selects Master Enable Register (described later), all the address signals A31-A24 at logic 0s, and all the address signals A23-A12 programmed for address matching in the above registers have to be at logic 0s. To define the lower memory area as 4 Kbytes, MA23-MA12 should be programmed with 1s. For an area larger than 4 Kbytes, MA23-MA12 (in that order) should be programmed with contiguous 1s followed by contiguous 0s. This is the intended usage to maintain the lower memory area as a single block. Note also that /LMCS can be enabled for refresh transactions independent of the value programmed into the Memory Selects Master Enable Register.

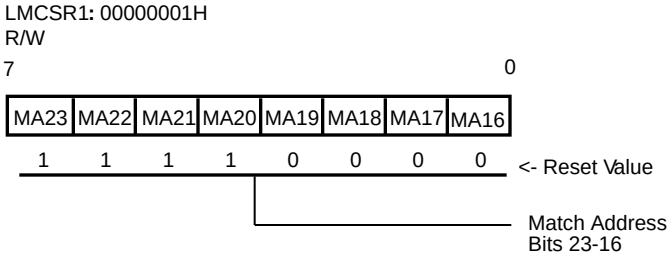

Figure 33. Lower Memory Chip Select Register 1

Mid-Range Memory Wait Register 2

T1W2-T1W0 *(T1 Waits).* This binary field defines up to seven T1 wait states to be inserted in transactions accessing the mid-range memory area 2 in chip select scheme 1.

T2W1-T2W0 *(T2 Waits).* This binary field defines up to three T2 wait states to be inserted in transactions accessing the mid-range memory area 2 in chip select scheme 1.

T3W2-T3W0 *(T3 Waits).* This binary field defines up to seven T3 wait states to be inserted in transactions accessing the mid-range memory area 2 in chip select scheme 1. The contents of this register have no effects in chip select scheme 2.

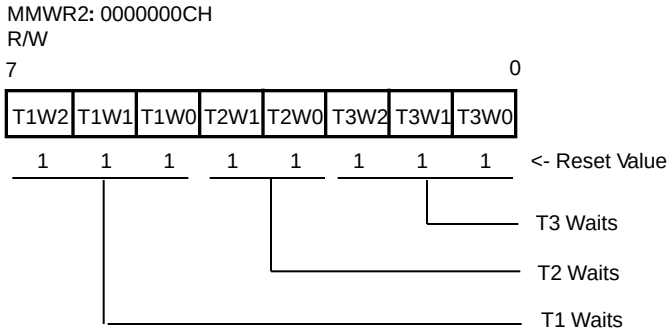

Figure 44. Mid-Range Memory Waits Register 2

Mid-range Memory Waits Register 3

T1W2-T1W0 *(T1 Waits).* This binary field defines up to seven T1 wait states to be inserted in transactions accessing the mid-range memory area 3 in chip select scheme 1.

T2W1-T2W0 *(T2 Waits).* This binary field defines up to three T2 wait states to be inserted in transactions accessing the mid-range memory area 3 in chip select scheme 1.

T3W2-T3W0 *(T3 Waits).* This binary field defines up to seven T3 wait states to be inserted in transactions accessing the mid-range memory area 3 in chip select scheme 1. The contents of this register have no effects in chip select scheme 2.

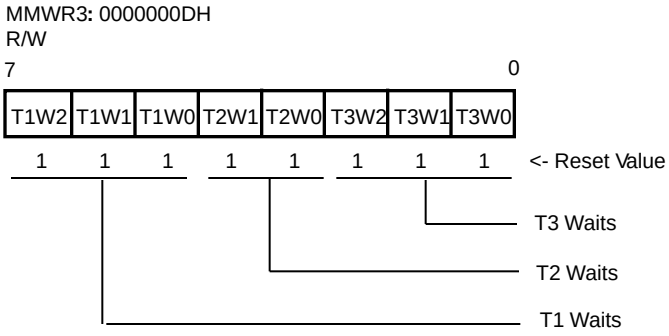

Figure 45. Mid-range Memory Waits Register 3

Standby Mode Exit With Bus Request

Optionally, if the BRXT bit of the Standby Mode Control Register (SMCR) was previously set, /STNBY goes to logic 1 when the /BREQ input is asserted, allowing the external crystal oscillator that drives the Z380 MPU's CLK input to restart. A warm-up counter internal to the Z380 MPU proceeds to count, for a duration long enough for the oscillator to stabilize, which was selected with the WM bits in the SMCR. When the counter reaches its end-count, clocking resumes within the Z380 MPU and at the BUSCLK and IOCLK outputs.

The Z380 MPU relinquishes the system bus after clocking resumes, with the normal /BREQ, /BACK handshake procedure. The Z380 MPU regains the system bus when /BREQ goes inactive, again going through a normal handshake procedure.

Note that clocking continues, and the Z380 MPU is at the halt state.

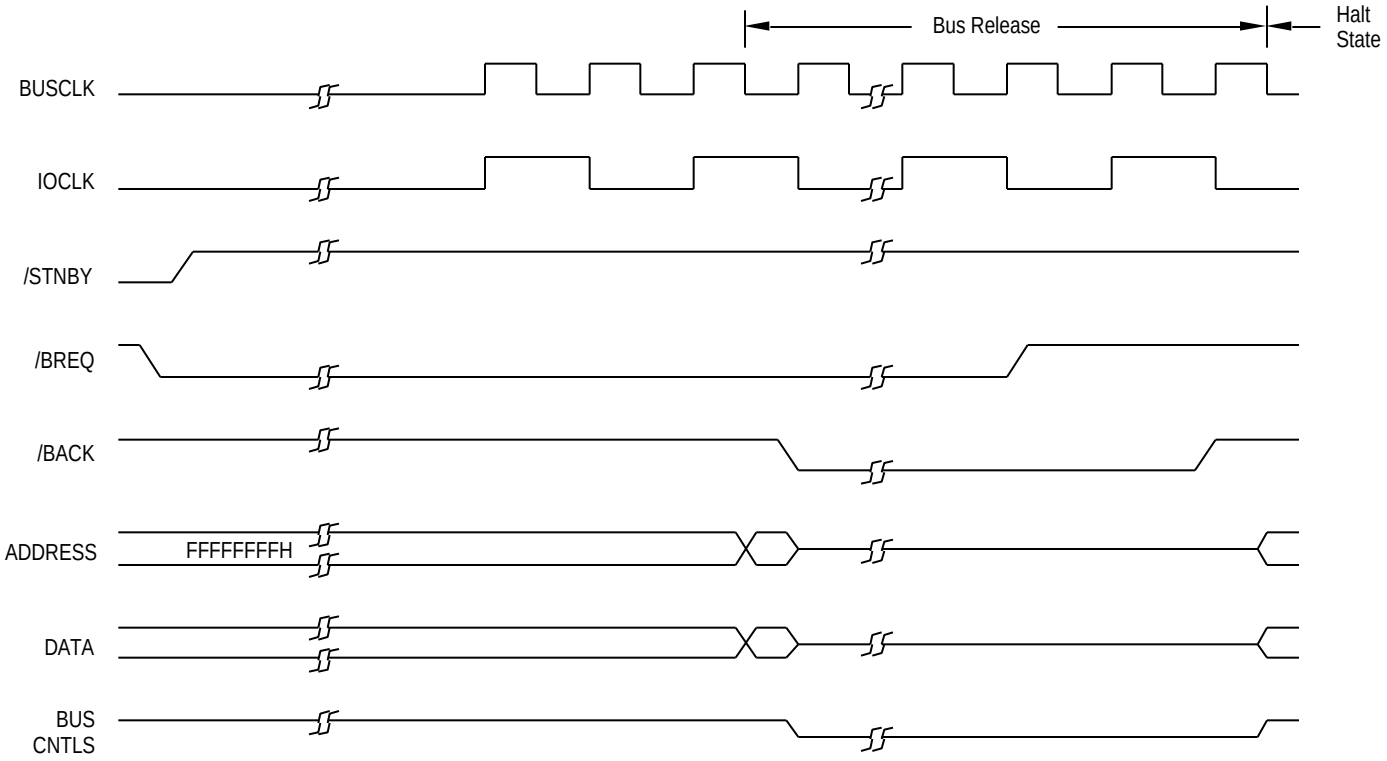

Figure 52. Standby Mode Exit with Bus Request Timing

No.	Symbol	Parameter	Z8L38010 Min	Z8L38010 Max	Note
1	TcC	CLK Cycle Time	100		
2	TwCh	CLK Width High	40		
3	TwCl	CLK Width Low	40		
4	TrC	CLK Rise Time		5	
5	TfC	CLK Fall Time		5	
6	TdCf(BCr)	CLK Fall to BUSCLK Rise Delay		60	
7	TdCr(BCf)	CLK Rise to BUSCLK Fall Delay		55	
8	TdBCr(OUT)	BUSCLK Rise to Output Valid Delay		15	
9	TdBCf(OUT)	BUSCLK Fall to Output Valid Delay		15	
10	TsIN(BCr)	Input to BUSCLK Rise Setup Time	30		1
11	ThIN(BCr)	Input to BUSCLK Rise Hold Time	0		1
12	TsBR(BCf)	/BREQ to BUSCLK Fall Setup Time	30		2
13	ThBR(BCf)	/BREQ to BUSCLK Fall Hold Time	0		2
14	TsMW(BCr)	Mem Wait to BUSCLK Rise Setup Time	30		3
15	ThMW(BCr)	Mem Wait to BUSCLK Rise Hold Time	0		3
16	TsMW(BCf)	Mem Wait to BUSCLK Fall Setup Time	45		3
17	ThMW(BCf)	Mem Wait to BUSCLK Fall Hold Time	0		3
18	TsIOW(BCr)	IO Wait to BUSCLK Rise Setup Time	45		3
19	ThIOW(BCr)	IO Wait to BUSCLK Rise Hold Time	0		3
20	TsIOW(BCf)	IO Wait to BUSCLK Fall Setup Time	45		3
21	ThIOW(BCf)	IO Wait to BUSCLK Fall Hold Time	0		3
22	TwNMI1	/NMI Low Width	50		
23	TwRES1	Reset Low Width	10		
24	Tx01(02)	Output Skew (Same Clock Edge)	–4	+4	4
25	Tx01(03)	Output Skew (Opposite Clock Edge)	–6	+6	5

Notes:
1. Applicable for Data Bus and /MSIZE inputs
2. /BREQ can also be asserted/deasserted asynchronously
3. External waits asserted at /WAIT input
4. Tx01(02) = [Output 1] TdBCr(OUT) - [Output 2] TdBCr(OUT)
   or [Output 1] TdBCf(OUT) - [Output 2] TdBCf(OUT)
5. Tx01(03) = [Output 1] TdBCr(OUT) - [Output 3] TdBCf(OUT)
   or [Output 1] TdBCf(OUT) - [Output 3] TdBCr(OUT)

	no esc	ED esc	DD esc	FD esc	CB esc	ED-CB	DD-CB	FD-CB
66	LD H,(HL)	-	LD H,(IX+d)	LD H,(IY+d)	BIT 4,(HL)	-	BIT 4,(IX+d)	BIT 4,(IY+d)
67	LD H,A	RRD	LD IXU,A	LD IYU,A	BIT 4,A	-	-	-
68	LD L,B	IN L,(C)	LD IXL,B	LD IYL,B	BIT 5,B	-	-	-
69	LD L,C	OUT (C),L	LD IXL,C	LD IYL,C	BIT 5,C	-	-	-
6A	LD L,D	ADC HL,HL	LD IXL,D	LD IYL,D	BIT 5,D	-	-	-
6B	LD L,E	LD HL,(nn)	LD IXL,E	LD IYL,E	BIT 5,E	-	-	-
6C	LD L,H	MLT HL	LD IXL,IXU	LD IYL,IYU	BIT 5,H	-	-	-
6D	LD L,L	-	LD IXL,IXL	LD IYL,IYL	BIT 5,L	-	-	-
6E	LD L,(HL)	-	LD L,(IX+d)	LD L,(IY+d)	BIT 5,(HL)	-	BIT 5,(IX+d)	BIT5,(IY+d)
6F	LD L,A	RLD	LD IXL,A	LD IYL,A	BIT 5,A	-	-	-
70	LD (HL),B	-	LD (IX+d),B	LD (IY+d),B	BIT 6,B	-	-	-
71	LD (HL),C	**OUT (C),n**	LD (IX+d),C	LD (IY+d),C	BIT 6,C	-	-	-
72	LD (HL),D	SBC HL,SP	LD (IX+d),D	LD (IY+d),D	BIT 6,D	-	-	-
73	LD (HL),E	LD (nn),SP	LD (IX+d),E	LD (IY+d),E	BIT 6,E	-	-	-
74	LD (HL),H	TSTIO m	LD (IX+d),H	LD (IY+d),H	BIT 6,H	-	-	-
75	LD (HL),L	**EXTSW**	LD (IX+d),L	LD (IY+d),L	BIT 6,L	-	-	-
76	HALT	SLP	-	-	BIT 6,(HL)	-	BIT 6,(IX+d)	BIT 6,(IY+d)
77	LD (HL),A	-	LD (IX+d),A	LD (IY+d),A	BIT 6,A	-	-	-
78	LD A,B	IN A,(C)	**INW HL,(C)**	-	BIT 7,B	-	-	-
79	LD A,C	OUT (C),A	**OUTW (C),HL**	**OUTW (C),nn**	BIT 7,C	-	-	-
7A	LD A,D	ADC HL,SP	-	-	BIT 7,D	-	-	-
7B	LD A,E	LD SP,(nn)	-	-	BIT 7,E	-	-	-
7C	LD A,H	MLT SP	LD A,IXU	LD A,IYU	BIT 7,H	-	-	-
7D	LD A,L	-	LD A,IXL	LD A,IYL	BIT 7,L	-	-	-
7E	LD A,(HL)	-	LD A,(IX+d)	LD A,(IY+d)	BIT 7,(HL)	-	BIT 7,(IX+d)	BIT 7,(IY+d)
7F	LD A,A	-	-	-	BIT 7,A	-	-	-
80	ADD A,B	-	-	-	RES 0,B	-	-	-
81	ADD A,C	-	-	-	RES 0,C	-	-	-
82	ADD A,D	**ADD SP,nn ****	-	-	RES 0,D	-	-	-
83	ADD A,E	OTIM	-	-	RES 0,E	-	-	-
84	ADD A,H	**ADDW BC**	ADD IXU	ADD IYU	RES 0,H	-	-	-
85	ADD A,L	**ADDW DE**	ADD IXL	ADD IYL	RES 0,L	-	-	-
86	ADD A,(HL)	**ADDW nn**	ADD A,(IX+d)	ADD A,(IY+d)	RES 0,(HL)	-	RES 0,(IX+d)	RES 0,(IY+d)
87	ADD A,A	**ADDW HL**	**ADDW IX**	**ADDW IY**	RES 0,A	-	-	-
88	ADC A,B	-	-	-	RES 1,B	-	-	-
89	ADC A,C	-	-	-	RES 1,C	-	-	-
8A	ADC A,D	-	-	-	RES 1,D	-	-	-
8B	ADC A,E	OTDM	-	-	RES 1,E	-	-	-
8C	ADC A,H	**ADCW BC**	ADC A,IXU	ADC A,IYU	RES 1,H	-	-	-
8D	ADC A,L	**ADCW DE**	ADC A,IXL	ADC A,IYL	RES 1,L	-	-	-
8E	ADC A,(HL)	**ADCW nn**	ADC A,(IX+d)	ADC A,(IY+d)	RES 1,(HL)	-	RES 1,(IX+d)	RES 1,(IY+d)
8F	ADC A,A	**ADCW HL**	**ADCW IX**	**ADCW IY**	RES 1,A	-	-	-
90	SUB B	-	-	-	RES 2,B	**MULTW BC**	-	-
91	SUB C	-	-	-	RES 2,C	**MULTW DE**	-	-
92	SUB D	**SUB SP,nn ****	-	-	RES 2,D	-	**MULTW (IX+d)**	**MULTW (IY+d)**
93	SUB E	OTIMR	-	-	RES 2,E	**MULTW HL**	-	-
94	SUB H	SUBW BC	SUB IXU	SUB IYU	RES 2,H	**MULTW IX**	-	-
95	SUB L	SUBW DE	SUB IXL	SUB IYL	RES 2,L	**MULTW IY**	-	-
96	SUB (HL)	SUBW nn	SUB (IX+d)	SUB (IY+d)	RES 2,(HL)	-	RES 2,(IX+d)	RES 2,(IY+d)
97	SUB A	**SUBW HL**	**SUBW IX**	**SUBW IY**	RES 2,A	**MULTW nn**	-	-

	no esc	ED esc	DD esc	FD esc	CB esc	ED-CB	DD-CB	FD-CB
CA	JP Z,nn	-	**LDCTL SR,n**	-	SET 1,D	-	-	-
CB	escape	escape	escape	escape	SET 1,E	-	-	-
CC	CALL Z,nn	**CALR Z,e**	**CALR Z,ee**	**CALR Z,eee**	SET 1,H	-	-	-
CD	CALL nn	**CALR e**	**CALR ee**	**CALR eee**	SET 1,L	-	-	-
CE	ADC A,n	-	**ADCW (IX+d)**	**ADCW (IY+d)**	SET 1,(HL)	-	SET 1,(IX+d)	SET 1,(IY+d)
CF	RST 1	**BTEST**	**MTEST**	-	SET 1,A	-	-	-
D0	RET NC	**LDCTL A,DSR**	**LDCTL A,XSR**	**LDCTL A,YSR**	SET 2,B	-	-	-
D1	POP DE	-	-	-	SET 2,C	-	-	-
D2	JP NC,nn	-	-	-	SET 2,D	-	-	-
D3	OUT (n),A	**OUTA (nn),A**	-	**OUTAW (nn),HL**	SET 2,E	-	-	-
D4	CALL NC,nn	**CALR NC,e**	**CALR NC,ee**	**CALR NC,eee**	SET 2,H	-	-	-
D5	PUSH DE	-	-	-	SET 2,L	-	-	-
D6	SUB n	**SUB HL,(nn) ****	**SUBW (IX+d)**	**SUBW (IY+d)**	SET 2,(HL)	-	SET 2,(IX+d)	SET 2,(IY+d)
D7	RST 2	-	-	-	SET 2,A	-	-	-
D8	RET C	**LDCTL DSR,A**	**LDCTL XSR,A**	**LDCTL YSR,A**	SET 3,B	-	-	-
D9	EXX	**EXALL**	**EXXX**	**EXXY**	SET 3,C	-	-	-
DA	JP C,nn	**LDCTL DSR,n**	**LDCTL XSR,n**	**LDCTL YSR,n**	SET 3,D	-	-	-
DB	IN A,(n)	**INA A,(nn)**	-	**INAW HL,(nn)**	SET 3,E	-	-	-
DC	CALL C,nn	**CALR C,e**	**CALR C,ee**	**CALR C,eee**	SET 3,H	-	-	-
DD	escape	reserved	reserved	reserved	SET 3,L	-	-	-
DE	SBC A,n	-	**SBCW (IX+d)**	**SBCW (IY+d)**	SET 3,(HL)	-	SET 3,(IX+d)	SET 3,(IY+d)
DF	RST 3	-	-	-	SET 3,A	-	-	-
E0	RET PO	**LDIW**	-	-	SET 4,B	-	-	-
E1	POP HL	-	POP IX	POP IY	SET 4,C	-	-	-
E2	JP PO,nn	**INIW**	-	-	SET 4,D	-	-	-
E3	EX (SP),HL	**OUTIW**	EX (SP),IX	EX (SP),IY	SET 4,E	-	-	-
E4	CALL PO,nn	**CALR PO,e**	**CALR PO,ee**	**CALR PO,eee**	SET 4,H	-	-	-
E5	PUSH HL	-	PUSH IX	PUSH IY	SET 4,L	-	-	-
E6	AND n	-	**ANDW (IX+d)**	**ANDW (IY+d)**	SET 4,(HL)	-	SET 4,(IX+d)	SET 4,(IY+d)
E7	RST 4	-	-	-	SET 4,A	-	-	-
E8	RET PE	**LDDW**	-	-	SET 5,B	-	-	-
E9	JP (HL)	-	JP (IX)	JP (IY)	SET 5,C	-	-	-
EA	JP PE,nn	**INDW**	-	-	SET 5,D	-	-	-
EB	EX DE,HL	**OUTDW**	-	-	SET 5,E	-	-	-
EC	CALL PE,nn	**CALR PE,e**	**CALR PE,ee**	**CALR PE,eee**	SET 5,H	-	-	-
ED	escape	reserved	reserved	reserved	SET 5,L	-	-	-
EE	XOR n	-	**XORW (IX+d)**	**XORW (IY+d)**	SET 5,(HL)	-	SET 5,(IX+d)	SET 5,(IY+d)
EF	RST 5	-	-	-	SET 5,A	-	-	-
F0	RET P	**LDIRW**	-	-	SET 6,B	-	-	-
F1	POP AF	-	-	-	SET 6,C	-	-	-
F2	JP P,nn	**INIRW**	-	-	SET 6,D	-	-	-
F3	DI	**OTIRW**	**DI n**	-	SET 6,E	-	-	-
F4	CALL P,nn	**CALR P,e**	**CALR P,ee**	**CALR P,eee**	SET 6,H	-	-	-
F5	PUSH AF	-	-	**PUSH nn**	SET 6,L	-	-	-
F6	OR n	-	**ORW (IX+d)**	**ORW (IY+d)**	SET 6,(HL)	-	SET 6,(IX+d)	SET 6,(IY+d)
F7	RST 6	**SETC LCK**	**SETC LW**	**SETC XM**	SET 6,A	-	-	-
F8	RET M	**LDDRW**	-	-	SET 7,B	-	-	-
F9	LD SP,HL	-	LD SP,IX	LD SP,IY	SET 7,C	-	-	-
FA	JP M,nn	**INDRW**	-	-	SET 7,D	-	-	-
FB	EI	**OTDRW**	**EI n**	-	SET 7,E	-	-	-
FC	CALL M,nn	**CALR M,e**	**CALR M,ee**	**CALR M,eee**	SET 7,H	-	-	-
FD	escape	reserved	reserved	reserved	SET 7,L	-	-	-
FE	CP n	-	**CPW (IX+d)**	**CPW (IY+d)**	SET 7,(HL)	-	SET 7,(IX+d)	SET 7,(IY+d)
FF	RST 7	**RESC LCK**	**RESC LW**	**-**	SET 7,A	-	-	-