National Semiconductor

microCMOS

NSC800™ High-Performance Low-Power CMOS Microprocessor

General Description

The NSC800 is an 8-bit CMOS microprocessor that functions as the central processing unit (CPU) in National Semiconductor's NSC800 microcomputer family. National's microCMOS technology used to fabricate this device provides system designers with performance equivalent to comparable NMOS products, but with the low power advantage of CMOS. Some of the many system functions incorporated on the device, are vectored priority interrupts, refresh control, power-save feature and interrupt acknowledge. The NSC800 is available in dual-in-line and surface mounted chip carrier packages.

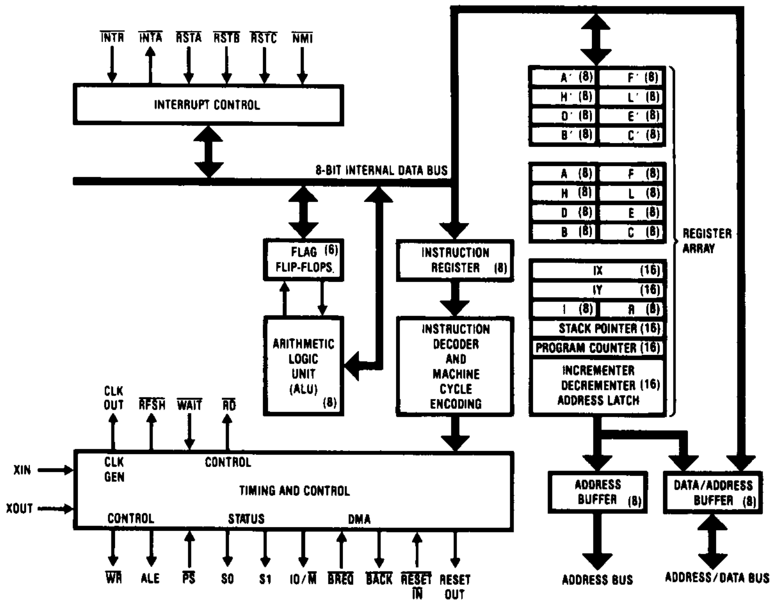
The system designer can choose not only from the dedicated CMOS peripherals that allow direct interfacing to the NSC800 but from the full line of National's CMOS products to allow a low-power system solution. The dedicated peripherals include NSC810A RAM I/O Timer, NSC858 UART, and NSC831 I/O.

All devices are available in commercial, industrial and military temperature ranges along with two added reliability flows. The first is an extended burn in test and the second is the military class C screening in accordance with Method 5004 of MIL-STD-883.

Features

- Fully compatible with Z80® instruction set:
  Powerful set of 158 instructions
  10 addressing modes
  22 internal registers
- Low power: 50 mW at 5V V_{CC}
- Unique power-save feature
- Multiplexed bus structure
- Schmitt trigger input on reset
- On-chip bus controller and clock generator
- Variable power supply 2.4V−6.0V
- On-chip 8-bit dynamic RAM refresh circuitry
- Speed: 1.0 μs instruction cycle at 4.0 MHz
  NSC800-4 4.0 MHz
  NSC800-35 3.5 MHz
  NSC800-3 2.5 MHz
  NSC800-1 1.0 MHz
- Capable of addressing 64k bytes of memory and 256 I/O devices
- Five interrupt request lines on-chip

Block Diagram



TL/C/5171-73

Table of Contents

1.0 Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Storage Temperature	−65°C to +150°C
Voltage on Any Pin with Respect to Ground	−0.3V to V_{CC} +0.3V
Maximum V_{CC}	7V
Power Dissipation	1W
Lead Temp. (Soldering, 10 seconds)	300°C

2.0 Operating Conditions

NSC800-1 → T_A = 0°C to +70°C
T_A = −40°C to +85°C
NSC800-3 → T_A = 0°C to +70°C
T_A = −40°C to +85°C
T_A = −55°C to +125°C
NSC800-35/883C → T_A = −55°C to +125°C
NSC800-4 → T_A = 0°C to +70°C
T_A = −40°C to +85°C
NSC800-4MIL → T_A = −55°C to +90°C

3.0 DC Electrical Characteristics V_{CC} = 5V ±10%, GND = 0V, unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Logical 1 Input Voltage		0.8 V_{CC}		V_{CC}	V
V_{IL}	Logical 0 Input Voltage		0		0.2 V_{CC}	V
V_{HY}	Hysteresis at RESET IN input	V_{CC} = 5V	0.25	0.5		V
V_{OH1}	Logical 1 Output Voltage	I_{OUT} = −1.0 mA	2.4			V
V_{OH2}	Logical 1 Output Voltage	I_{OUT} = −10 μA	V_{CC} −0.5			V
V_{OL1}	Logical 0 Output Voltage	I_{OUT} = 2 mA	0		0.4	V
V_{OL2}	Logical 0 Output Voltage	I_{OUT} = 10 μA	0		0.1	V
I_{IL}	Input Leakage Current	0 ≤ V_{IN} ≤ V_{CC}	−10.0		10.0	μA
I_{OL}	Output Leakage Current	0 ≤ V_{IN} ≤ V_{CC}	−10.0		10.0	μA
I_{CC}	Active Supply Current	I_{OUT} = 0, $f_{(XIN)}$ = 2 MHz, T_A = 25°C		8	11	mA
I_{CC}	Active Supply Current	I_{OUT} = 0, $f_{(XIN)}$ = 5 MHz, T_A = 25°C		10	15	mA
I_{CC}	Active Supply Current	I_{OUT} = 0, $f_{(XIN)}$ = 7 MHz, T_A = 25°C		15	21	mA
I_{CC}	Active Supply Current	I_{OUT} = 0, $f_{(XIN)}$ = 8 MHz, T_A = 25°C		15	21	mA
I_Q	Quiescent Current	I_{OUT} = 0, $\overline{PS}$ = 0, V_{IN} = 0 or V_{IN} = V_{CC} $f_{(XIN)}$ = 0 MHz, T_A = 25°C, X_{IN} = 0, CLK = 1		2	5	mA
I_{PS}	Power-Save Current	I_{OUT} = 0, $\overline{PS}$ = 0, V_{IN} = 0 or V_{IN} = V_{CC} $f_{(XIN)}$ = 5.0 MHz, T_A = 25°		5	7	mA
C_{IN}	Input Capacitance			6	10	pF
C_{OUT}	Output Capacitance			8	12	pF
V_{CC}	Power Supply Voltage	(Note 2)	2.4	5	6	V

Note 1: Absolute Maximum Ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under DC Electrical Characteristics.

Note 2: CPU operation at lower voltages will reduce the maximum operating speed. Operation at voltages other than 5V ±10% is guaranteed by design, not tested.

4.0 AC Electrical Characteristics V_{CC} = 5V ±10%, GND = 0V, unless otherwise specified

Symbol	Parameter	NSC800-1		NSC800-3		NSC800-35		NSC800-4		Units	Notes
		Min	Max	Min	Max	Min	Max	Min	Max		
t_X	Period at XIN and XOUT Pins	500	3333	200	3333	142	3333	125	3333	ns	
T	Period at Clock Output (=2 t_X)	1000	6667	400	6667	284	6667	250	6667	ns	
t_R	Clock Rise Time		110		110		90		80	ns	Measured from 10%–90% of signal
t_F	Clock Fall Time		70		60		55		50	ns	Measured from 10%–90% of signal
t_L	Clock Low Time	435		150		90		80		ns	50% duty cycle, square wave input on XIN
t_H	Clock High Time	450		145		85		75		ns	50% duty cycle, square wave input on XIN
$t_{ACC(OP)}$	ALE to Valid Data		1340		490		340		300	ns	Add t for each $\overline{\text{WAIT}}$ STATE
$t_{ACC(MR)}$	ALE to Valid Data		1875		620		405		360	ns	Add t for each $\overline{\text{WAIT}}$ STATE
t_{AFR}	AD(0–7) Float after $\overline{\text{RD}}$ Falling		0		0		0		0	ns	
t_{BABE}	$\overline{\text{BACK}}$ Rising to Bus Enable		1000		400		300		250	ns	
t_{BABF}	$\overline{\text{BACK}}$ Falling to Bus Float		50		50		50		50	ns	
t_{BACL}	$\overline{\text{BACK}}$ Fall to CLK Falling	425		125		60		55		ns	
t_{BRH}	$\overline{\text{BREQ}}$ Hold Time	0		0		0		0		ns	
t_{BRS}	$\overline{\text{BREQ}}$ Set-Up Time	100		50		50		45		ns	
t_{CAF}	Clock Falling ALE Falling	0	70	0	65	0	60	0	55	ns	
t_{CAR}	Clock Rising to ALE Rising	0	100	0	100	0	90	0	80	ns	
t_{CRD}	Clock Rising to Read Rising		100		90		90		80	ns	
t_{CRF}	Clock Rising to Refresh Falling		80		70		70		65	ns	
t_{DAI}	ALE Falling to $\overline{\text{INTA}}$ Falling	445		160		95		85		ns	
t_{DAR}	ALE Falling to $\overline{\text{RD}}$ Falling	400	575	160	250	100	180	90	160	ns	
t_{DAW}	ALE Falling to $\overline{\text{WR}}$ Falling	900	1010	350	420	225	300	200	265	ns	
$t_{D(BACK)1}$	ALE Falling to $\overline{\text{BACK}}$ Falling	2460		975		635		560		ns	Add t for each $\overline{\text{WAIT}}$ state Add t for opcode fetch cycles
$t_{D(BACK)2}$	$\overline{\text{BREQ}}$ Rising to $\overline{\text{BACK}}$ Rising	500	1610	200	700	140	540	125	475	ns	
$t_{D(I)}$	ALE Falling to $\overline{\text{INTR}}$, $\overline{\text{NMI}}$, $\overline{\text{RSTA-C}}$, $\overline{\text{PS}}$, $\overline{\text{BREQ}}$, Inputs Valid		1360		475		284		250	ns	Add t for each $\overline{\text{WAIT}}$ state Add t for opcode fetch cycles
t_{DPA}	Rising $\overline{\text{PS}}$ to Falling ALE	500	1685	200	760	140	580	125	510	ns	See *Figure 14* also
$t_{D(WAIT)}$	ALE Falling to $\overline{\text{WAIT}}$ Input Valid		550		250		170		125	ns	

OP— Opcode Fetch
MR— Memory Read

4.0 AC Electrical Characteristics V_{CC} = 5V ±10%, GND = 0V, unless otherwise specified (Continued)

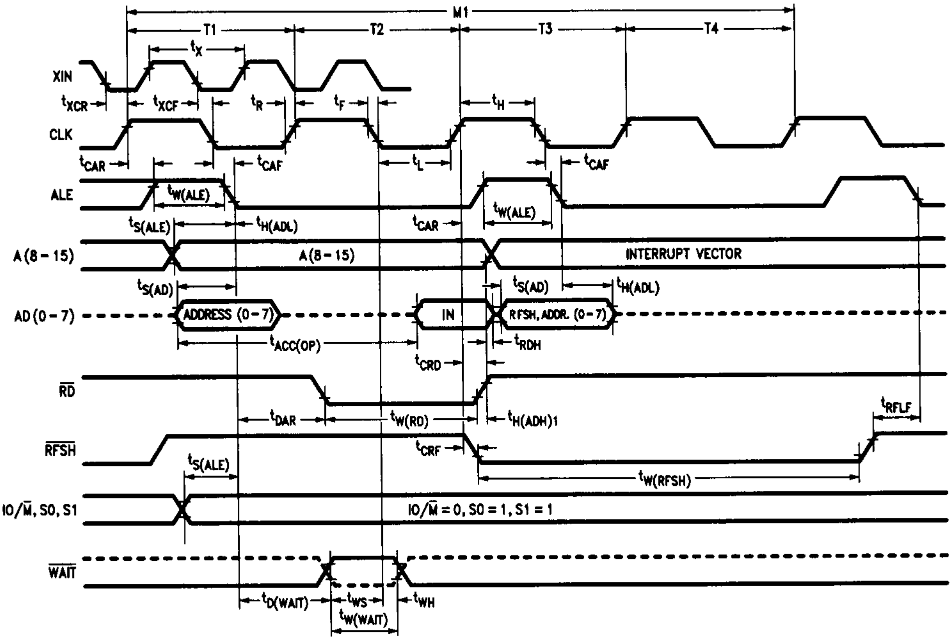
Symbol	Parameter	NSC800-1		NSC800-3		NSC800-35		NSC800-4		Units	Notes
		Min	Max	Min	Max	Min	Max	Min	Max		
$T_{H(ADH)1}$	A(8–15) Hold Time During Opcode Fetch	0		0		0		0		ns	
$T_{H(ADH)2}$	A(8–15) Hold Time During Memory or IO, $\overline{RD}$ and $\overline{WR}$	400		100		85		60		ns	
$T_{H(ADL)}$	AD(0–7) Hold Time	100		60		35		30		ns	
$T_{H(WD)}$	Write Data Hold Time	400		100		85		75		ns	
t_{INH}	Interrupt Hold Time	0		0		0		0		ns	
t_{INS}	Interrupt Set-Up Time	100		50		50		45		ns	
t_{NMI}	Width of NMI Input	50		30		25		20		ns	
t_{RDH}	Data Hold after Read	0		0		0		0		ns	
t_{RFLF}	$\overline{RFSH}$ Rising to ALE Falling	60		50		45		40		ns	
$t_{RL(MR)}$	$\overline{RD}$ Rising to ALE Rising (Memory Read)	390		100		50		45		ns	
$t_{S(AD)}$	AD(0–7) Set-Up Time	300		45		45		40		ns	
$t_{S(ALE)}$	A(8–15), SO, SI, IO/$\overline{M}$ Set-Up Time	350		70		55		50		ns	
$t_{S(WD)}$	Write Data Set-Up Time	385		75		35		30		ns	
$t_{W(ALE)}$	ALE Width	430		130		115		100		ns	
t_{WH}	$\overline{WAIT}$ Hold Time	0		0		0		0		ns	
$t_{W(I)}$	Width of $\overline{INTR}$, $\overline{RSTA\text{-}C}$, $\overline{PS}$, $\overline{BREQ}$	500		200		140		125		ns	
$t_{W(INTA)}$	$\overline{INTA}$ Strobe Width	1000		400		225		200		ns	Add two t states for first $\overline{INTA}$ of each interrupt response string Add t for each $\overline{WAIT}$ state
t_{WL}	$\overline{WR}$ Rising to ALE Rising	450		130		70		70		ns	
$t_{W(RD)}$	Read Strobe Width During Opcode Fetch	960		360		210		185		ns	Add t for each $\overline{WAIT}$ State Add t/2 for Memory Read Cycles
$t_{W(RFSH)}$	Refresh Strobe Width	1925		725		450		395		ns	
t_{WS}	$\overline{WAIT}$ Set-Up Time	100		70		60		55		ns	
$t_{W(WAIT)}$	$\overline{WAIT}$ Input Width	550		250		195		175		ns	
$t_{W(WR)}$	Write Strobe Width	985		370		250		220		ns	Add t for each $\overline{WAIT}$ state
t_{XCF}	XIN to Clock Falling	25	100	15	85	5	90	5	80	ns	
t_{XCR}	XIN to Clock Rising	25	85	15	85	5	90	5	80	ns	

Note 1: Test conditions: t = 1000 ns for NSC800-1, 400 ns for NSC800, 285 ns for NSC800-35, 250 ns for NSC800-4.

Note 2: Output timings are measured with a purely capacitive load of 100 pF.

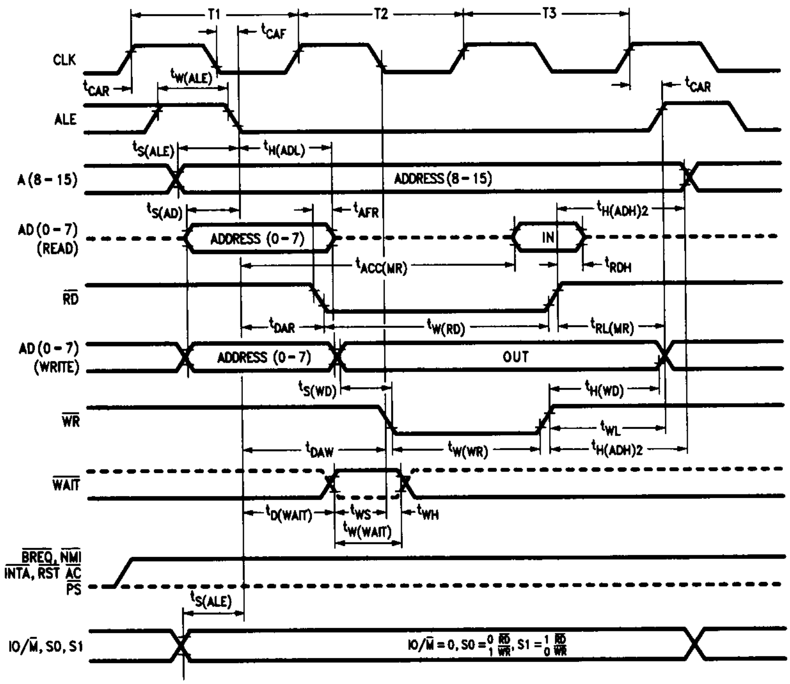
5.0 Timing Waveforms

Opcode Fetch Cycle



TL/C/5171-3

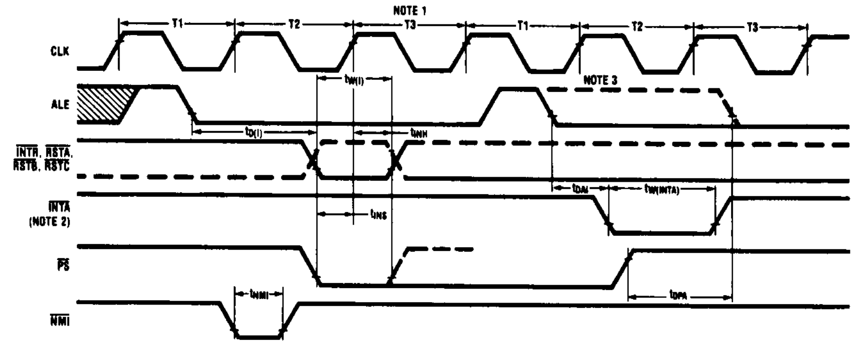
Memory Read and Write Cycle



TL/C/5171-4

5.0 Timing Waveforms (Continued)

Interrupt—Power-Save Cycle



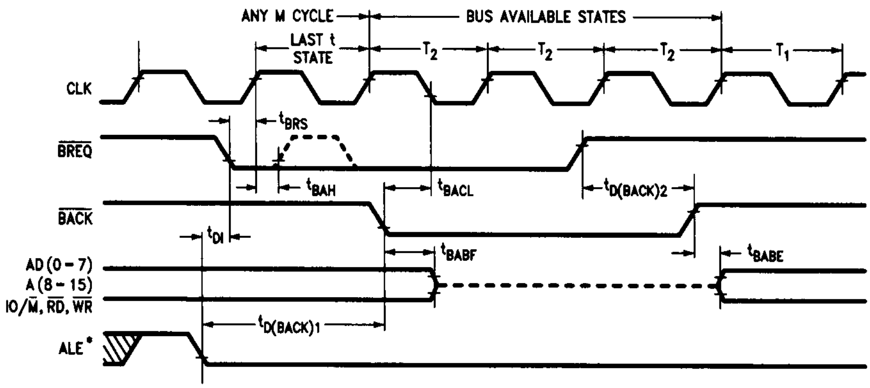
TL/C/5171–5

Note 1: This t state is the last t state of the last M cycle of any instruction.

Note 2: Response to INTR input.

Note 3: Response to PS input.

Bus Acknowledge Cycle



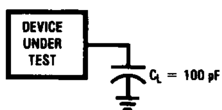
TL/C/5171–6

*Waveform not drawn to proportion. Use only for specifying test points.

AC Testing Input/Output Waveform

TL/C/5171–7

AC Testing Load Circuit



TL/C/5171–8

7

NSC800 HARDWARE

6.0 Pin Descriptions

6.1 INPUT SIGNALS

Reset Input (RESET IN): Active low. Sets A (8–15) and AD (0–7) to TRI-STATE® (high impedance). Clears the contents of PC, I and R registers, disables interrupts, and activates reset out.

Bus Request (BREQ): Active low. Used when another device requests the system bus. The NSC800 recognizes BREQ at the end of the current machine cycle, and sets A(8–15), AD(0–7), IO/M, RD, and WR to the high impedance state. RFSH is high during a bus request cycle. The CPU acknowledges the bus request via the BACK output signal.

Non-Maskable Interrupt (NMI): Active low. The non-maskable interrupt, generated by the peripheral device(s), is the highest priority interrupt. The edge sensitive interrupt requires only a pulse to set an internal flip-flop which generates the internal interrupt request. The NMI flip-flop is monitored on the same clock edge as the other interrupts. It must also meet the minimum set-up time spec for the interrupt to be accepted in the current machine instruction. When the processor accepts the interrupt the flip-flop resets automatically. Interrupt execution is independent of the interrupt enable flip-flop. NMI execution results in saving the PC on the stack and automatic branching to restart address X'0066 in memory.

Restart Interrupts, A, B, C (RSTA, RSTB, RSTC): Active low level sensitive. The CPU recognizes restarts generated by the peripherals at the end of the current instruction, if their respective interrupt enable and master enable bits are set. Execution is identical to NMI except the interrupts vector to the following restart addresses:

Name	Restart Address (X')
NMI	0066
RSTA	003C
RSTB	0034
RSTC	002C
INTR (Mode 1)	0038

The order of priority is fixed. The list above starts with the highest priority.

Interrupt Request (INTR): Active low, level sensitive. The CPU recognizes an interrupt request at the end of the current instruction provided that the interrupt enable and master interrupt enable bits are set. INTR is the lowest priority interrupt. Program control selects one of three response modes which determines the method of servicing INTR in conjunction with INTA. See Interrupt Control.

Wait (WAIT): Active low. When set low during RD, WR or INTA machine cycles (during the WR machine cycle, wait must be valid prior to write going active) the CPU extends its machine cycle in increments of t (wait) states. The wait machine cycle continues until the WAIT input returns high.

The wait strobe input will be accepted only during machine cycles that have RD, WR or INTA strobes and during the machine cycle immediately after an interrupt has been accepted by the CPU. The later cycle has its RD strobe suppressed but it will still accept the wait.

Power-Save (PS): Active low. PS is sampled during the last t state of the current instruction cycle. When PS is low, the CPU stops executing at the end of current instruction and keeps itself in the low-power mode. Normal operation resumes when PS returns high (see Power Save Feature description).

CRYSTAL (X_{IN}, X_{OUT}): X_{IN} can be used as an external clock input. A crystal can be connected across X_{IN} and X_{OUT} to provide a source for the system clock.

6.2 OUTPUT SIGNALS

Bus Acknowledge (BACK): Active low. BACK indicates to the bus requesting device that the CPU bus and its control signals are in the TRI-STATE mode. The requesting device then commands the bus and its control signals.

Address Bits 8–15 [A(8–15)]: Active high. These are the most significant 8 bits of the memory address during a memory instruction. During an I/O instruction, the port address on the lower 8 address bits gets duplicated onto A(8–15). During a BREQ/BACK cycle, the A(8–15) bus is in the TRI-STATE mode.

Reset Out (RESET OUT): Active high. When RESET OUT is high, it indicates the CPU is being reset. This signal is normally used to reset the peripheral devices.

Input/Output/Memory (IO/M): An active high on the IO/M output signifies that the current machine cycle is an input/output cycle. An active low on the IO/M output signifies that the current machine cycle is a memory cycle. It is TRI-STATE during BREQ/BACK cycles.

Refresh (RFSH): Active low. The refresh output indicates that the dynamic RAM refresh cycle is in progress. RFSH goes low during T3 and T4 states of all M1 cycles. During the refresh cycle, AD(0–7) has the refresh address and A(8–15) indicates the interrupt vector register data. RFSH is high during BREQ/BACK cycles.

Address Latch Enable (ALE): Active high. ALE is active only during the T1 state of any M cycle and also T3 state of the M1 cycle. The high to low transition of ALE indicates that a valid memory, I/O or refresh address is available on the AD(0–7) lines.

Read Strobe (RD): Active low. The CPU receives data via the AD(0–7) lines on the trailing edge of the RD strobe. The RD line is in the TRI-STATE mode during BREQ/BACK cycles.

Write Strobe (WR): Active low. The CPU sends data via the AD(0–7) lines while the WR strobe is low. The WR line is in the TRI-STATE mode during BREQ/BACK cycles.

Clock (CLK): CLK is the output provided for use as a system clock. The CLK output is a square wave at one half the input frequency.

Interrupt Acknowledge (INTA): Active low. This signal strobes the interrupt response vector from the interrupting peripheral devices onto the AD(0–7) lines. INTA is active during the M1 cycle immediately following the t state where the CPU recognized the INTR interrupt request.

Two of the three interrupt request modes use INTA. In mode 0 one to four INTA signals strobe a one to four byte instruction onto the AD(0–7) lines. In mode 2 one INTA signal strobes the lower byte of an interrupt response vector onto the bus. In mode 1, INTA is inactive and the CPU response to INTR is the same as for an NMI or restart interrupt.

6.0 Pin Descriptions (Continued)

Status (SO, S1): Bus status outputs provide encoded information regarding the current M cycle as follows:

Machine Cycle	Status			Control	
	S0	S1	IO/$\overline{M}$	$\overline{RD}$	$\overline{WR}$
Opcode Fetch	1	1	0	0	1
Memory Read	0	1	0	0	1
Memory Write	1	0	0	1	0
I/O Read	0	1	1	0	1
I/O Write	1	0	1	1	0
Halt*	0	0	0	0	1
Internal Operation*	0	1	0	1	1
Acknowledge of Int**	1	1	0	1	1

*ALE is not suppressed in this cycle.

**This is the cycle that occurs immediately after the CPU accepts an interrupt ($\overline{RSTA}$, $\overline{RSTB}$, $\overline{RSTC}$, $\overline{INTR}$, $\overline{NMI}$).

Note 1: During halt, CPU continues to do dummy opcode fetch from location following the halt instruction with a halt status. This is so CPU can continue to do its dynamic RAM refresh.

Note 2: No early status is provided for interrupt or hardware restarts.

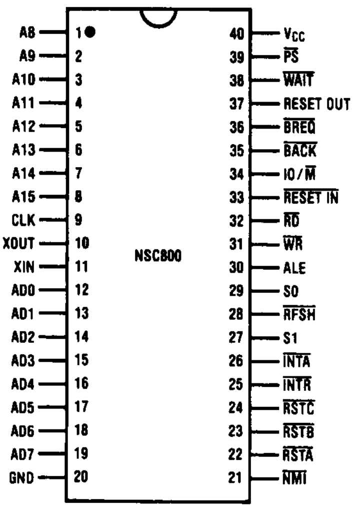
6.3 INPUT/OUTPUT SIGNALS

Multiplexed Address/Data [AD(0–7)]: Active high

At $\overline{RD}$ Time:	Input data to CPU.
At $\overline{WR}$ Time:	Output data from CPU.
At Falling Edge of ALE Time:	Least significant byte of address during memory reference cycle. 8-bit port address during I/O reference cycle.
During $\overline{BREQ}$/$\overline{BACK}$ Cycle:	High impedance.

7.0 Connection Diagrams

Dual-In-Line Package

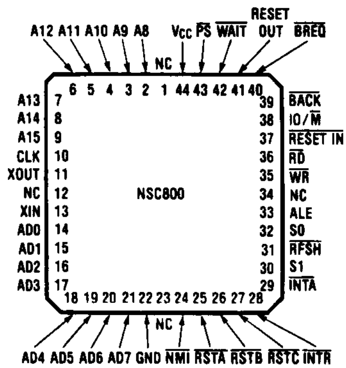


Top View

TL/C/5171-10

Order Number NSC800D or N
See NS Package D40C or N40A

Chip Carrier Package



Top View

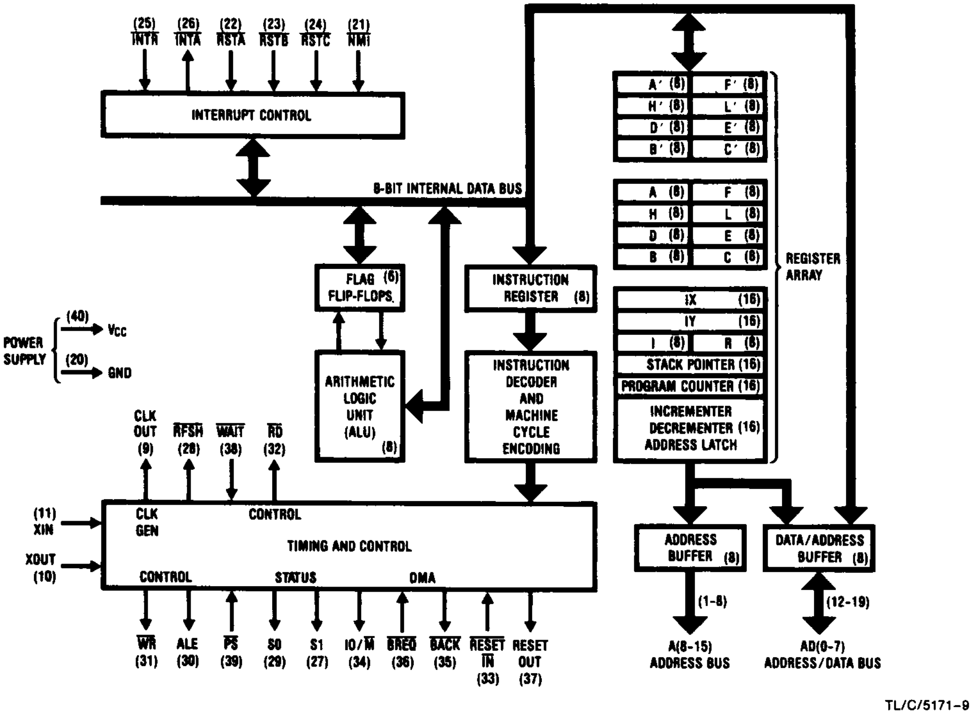
TL/C/5171-11

Order Number NSC800E or V
See NS Package E44B or V44A

8.0 Functional Description

This section reviews the CPU architecture shown below, focusing on the functional aspects from a hardware perspective, including timing details.

As illustrated in *Figure 1*, the NSC800 is an 8-bit parallel device. The major functional blocks are: the ALU, register array, interrupt control, timing and control logic. These areas are connected via the 8-bit internal data bus. Detailed descriptions of these blocks ae provided in the following sections.



Note: Applicable pinout for 40-pin dual-in-line package within parentheses

FIGURE 1. NSC800 CPU Functional Block Diagram

8.0 Functional Description (Continued)

8.1 REGISTER ARRAY

The NSC800 register array is divided into two parts: the dedicated registers and the working registers, as shown in *Figure 2*.

Main Reg. Set		Alternate Reg. Set		
Accumulator	**Flags**	**Accumulator**	**Flags**	
A	F	A′	F′	Working Registers
B	C	B′	C′	
D	E	D′	E′	
H	L	H′	L′	

Interrupt Vector I	Memory Refresh R	Dedicated Registers
Index Register IX		
Index Register IY		
Stack Pointer SP		
Program Counter PC		

FIGURE 2. NSC800 Register Array

8.2 DEDICATED REGISTERS

There are 6 dedicated registers in the NSC800: two 8-bit and four 16-bit registers (see *Figure 3*).

Although their contents are under program control, the program has no control over their operational functions, unlike the CPU working registers. The function of each dedicated register is described as follows:

CPU Dedicated Registers	
Program Counter PC	(16)
Stack Pointer SP	(16)
Index Register IX	(16)
Index Register IY	(16)
Interrupt Vector Register I	(8)
Memory Refresh Register R	(8)

FIGURE 3. Dedicated Registers

8.2.1 Program Counter (PC)

The program counter contains the 16-bit address of the current instruction being fetched from memory. The PC increments after its contents have been transferred to the address lines. When a program jump occurs, the PC receives the new address which overrides the incrementer.

There are many conditional and unconditional jumps, calls, and return instructions in the NSC800's instruction repertoire that allow easy manipulation of this register in controlling the program execution (i.e. JP NZ nn, JR Zd2, CALL NC, nn).

8.2.2 Stack Pointer (SP)

The 16-bit stack pointer contains the address of the current top of stack that is located in external system RAM. The stack is organized in a last-in, first-out (LIFO) structure. The pointer decrements before data is pushed onto the stack, and increments after data is popped from the stack.

Various operations store or retrieve, data on the stack. This, along with the usage of subroutine calls and interrupts, allows simple implementation of subroutine and interrupt nesting as well as alleviating many problems of data manipulation.

8.2.3 Index Register (IX and IY)

The NSC800 contains two index registers to hold independent, 16-bit base addresses used in the indexed addressing mode. In this mode, an index register, either IX or IY, contains a base address of an area in memory making it a pointer for data tables.

In all instructions employing indexed modes of operation, another byte acts as a signed two's complement displacement. This addressing mode enables easy data table manipulations.

8.2.4 Interrupt Register (I)

When the NSC800 provides a Mode 2 response to $\overline{\text{INTR}}$, the action taken is an indirect call to the memory location containing the service routine address. The pointer to the address of the service routine is formed by two bytes, the high-byte is from the I Register and the low-byte is from the interrupting peripheral. The peripheral always provides an even address for the lower byte (LSB = 0). When the processor receives the lower byte from the peripheral it concatenates it in the following manner:

I Register	External byte	
8 bits		0

↑ The LSB of the external byte must be zero.

FIGURE 4a. Interrupt Register

The even memory location contains the low-order byte, the next consecutive location contains the high-order byte of the pointer to the beginning address of the interrupt service routine.

8.2.5 Refresh Register (R)

For systems that use dynamic memories rather than static RAM's, the NSC800 provides an integral 8-bit memory refresh counter. The contents of the register are incremented after each opcode fetch and are sent out on the lower portion of the address bus, along with a refresh control signal. This provides a totally transparent refresh cycle and does not slow down CPU operation.

The program can read and write to the R register, although this is usually done only for test purposes.

8.0 Functional Description (Continued)

8.3 CPU WORKING AND ALTERNATE REGISTER SETS

8.3.1 CPU Working Registers

The portion of the register array shown in *Figure 4b* represents the CPU working registers. These sixteen 8-bit registers are general-purpose registers because they perform a multitude of functions, depending on the instruction being executed. They are grouped together also due to the types of instructions that use them, particularly alternate set operations.

The F (flag) register is a special-purpose register because its contents are more a result of machine status rather than program data. The F register is included because of its interaction with the A register, and its manipulations in the alternate register set operations.

8.3.2 Alternate Registers

The NSC800 registers designated as CPU working registers have one common feature: the existence of a duplicate register in an alternate register set. This architectural concept simplifies programming during operations such as interrupt response, when the machine status represented by the contents of the registers must be saved.

The alternate register concept makes one set of registers available to the programmer at any given time. Two instructions (EX AF, A'F' and EXX), exchange the current working set of registers with their alternate set. One exchange between the A and F registers and their respective duplicates (A' and F') saves the primary status information contained in the accumulator and the flag register. The second exchange instruction performs the exchange between the remaining registers, B, C, D, E, H, and L, and their respective alternates B', C', D', E', H', and L'. This essentially saves the contents of the original complement of registers while providing the programmer with a usable alternate set.

CPU Main Working Register Set

Accumulator A	(8)	Flags F	(8)
Register B	(8)	Register C	(8)
Register D	(8)	Register E	(8)
Register H	(8)	Register L	(8)

CPU Alternate Working Register Set

Accumulator A'	(8)	Flags F'	(8)
Register B'	(8)	Register C'	(8)
Register D'	(8)	Register E'	(8)
Register H'	(8)	Register L'	(8)

FIGURE 4b. CPU Working and Alternate Registers

8.4 REGISTER FUNCTIONS

8.4.1 Accumulator (A Register)

The A register serves as a source or destination register for data manipulation instructions. In addition, it serves as the accumulator for the results of 8-bit arithmetic and logic operations.

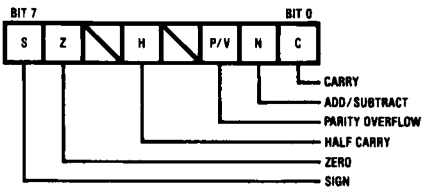
The A register also has a special status in some types of operations; that is, certain addressing modes are reserved for the A register only, although the function is available for all the other registers. For example, any register can be loaded by immediate, register indirect, or indexed addressing modes. The A register, however, can also be loaded via an additional register indirect addressing.

Another special feature of the A register is that it produces more efficient memory coding than equivalent instruction functions directed to other registers. Any register can be rotated; however, while it requires a two-byte instruction to normally rotate any register, a single-byte instruction is available for rotating the contents of the accumulator (A register).

8.4.2 F Register - Flags

The NSC800 flag register consists of six status bits that contain information regarding the results of previous CPU operations. The register can be read by pushing the contents onto the stack and then reading it, however, it cannot be written to. It is classified as a register because of its affiliation with the accumulator and the existence of a duplicate register for use in exchange instructions with the accumulator.

Of the six flags shown in *Figure 5*, only four can be directly tested by the programmer via conditional jump, call, and return instructions. They are the Sign (S), Zero (Z), Parity/Overflow (P/V), and Carry (C) flags. The Half Carry (H) and Add/Subtract (N) flags are used for internal operations related to BCD arithmetic.



TL/C/5171-23

FIGURE 5. Flag Register

8.0 Functional Description (Continued)

8.4.3 Carry (C)

A carry from the highest order bit of the accumulator during an add instruction, or a borrow generated during a subtraction instruction sets the carry flag. Specific shift and rotate instructions also affect this bit.

Two specific instructions in the NSC800 instruction repertoire set (SCF) or complement (CCF) the carry flag.

Other operations that affect the C flag are as follows:

- Adds
- Subtracts
- Logic Operations (always resets C flag)
- Rotate Accumulator
- Rotate and Shifts
- Decimal Adjust
- Negation of Accumulator

Other operations do not affect the C flag.

8.4.4 Adds/Subtract (N)

This flag is used in conjunction with the H flag to ensure that the proper BCD correction algorithm is used during the decimal adjust instruction (DAA). The correction algorithm depends on whether an add or subtract was previously done with BCD operands.

The operations that set the N flag are:

- Subtractions
- Decrements (8-bit)
- Complementing of the Accumulator
- Block I/O
- Block Searches
- Negation of the Accumulator

The operations that reset the N flag are:

- Adds
- Increments
- Logic Operations
- Rotates
- Set and Complement Carry
- Input Register Indirect
- Block Transfers
- Load of the I or R Registers
- Bit Tests

Other operations do not affect the N flag.

8.4.5 Parity/Overflow (P/V)

The Parity/Overflow flag is a dual-purpose flag that indicates results of logic and arithmetic operations. In logic operations, the P/V flag indicates the parity of the result; the flag is set (high) if the result is even, reset (low) if the result is odd. In arithmetic operations, it represents an overflow condition when the result, interpreted as signed two's complement arithmetic, is out of range for the eight-bit accumulator (i.e. -128 to $+127$).

The following operations affect the P/V flag according to the parity of the result of the operation:

- Logic Operations
- Rotate and Shift
- Rotate Digits
- Decimal Adjust
- Input Register Indirect

The following operations affect the P/V flag according to the overflow result of the operation.

- Adds (16 bit with carry, 8-bit with/without carry)
- Subtracts (16 bit with carry, 8-bit with/without carry)
- Increments and Decrements
- Negation of Accumulator

The P/V flag has no significance immediately after the following operations.

- Block I/O
- Bit Tests

In block transfers and compares, the P/V flag indicates the status of the BC register, always ending in the reset state after an auto repeat of a block move. Other operations do not affect the P/V flag.

8.4.6 Half Carry (H)

This flag indicates a BCD carry, or borrow, result from the low-order four bits of operation. It can be used to correct the results of a previously packed decimal add, or subtract, operation by use of the Decimal Adjust Instruction (DAA).

The following operations affect the H flag:

- Adds (8-bit)
- Subtracts (8-bit)
- Increments and Decrements
- Decimal Adjust
- Negation of Accumulator
- Always Set by: Logic AND
Complement Accumulator
Bit Testing
- Always Reset By: Logic OR's and XOR's
Rotates and Shifts
Set Carry
Input Register Indirect
Block Transfers
Loads of I and R Registers

The H flag has no significance immediately after the following operations.

- 16-bit Adds with/without carry
- 16-Bit Subtracts with carry
- Complement of the carry
- Block I/O
- Block Searches

Other operations do not affect the H flag.

8.0 Functional Description (Continued)

8.4.7 Zero Flag (Z)

Loading a zero in the accumulator or when a zero results from an operation sets the zero flag.

The following operations affect the zero flag.

- Adds (16-bit with carry, 8-bit with/without carry)
- Subtracts (16-bit with carry, 8-bit with/without carry)
- Logic Operations
- Increments and Decrements
- Rotate and Shifts
- Rotate Digits
- Decimal Adjust
- Input Register Indirect
- Block I/O (always set after auto repeat block I/O)
- Block Searches
- Load of I and R Registers
- Bit Tests
- Negation of Accumulator

The Z flag has no signficance immediately after the following operations:

- Block Transfers

Other operations do not affect the zero flag.

8.4.8 Sign Flag (S)

The sign flag stores the state of bit 7 (the most-significant bit and sign bit) of the accumulator following an arithmetic operation. This flag is of use when dealing with signed numbers.

The sign flag is affected by the following operation according to the result:

- Adds (16-bit with carry, 8-bit with/without carry)
- Subtracts (16-bit with carry, 8-bit with/without carry)
- Logic Operations
- Increments and Decrements
- Rotate and Shifts
- Rotate Digits
- Decimal Adjust
- Input Register Indirect
- Block Search
- Load of I and R Registers
- Negation of Accumulator

The S flag has no significance immediately after the following operations:

- Block I/O
- Block Transfers
- Bit Tests

Other operations do not affect the sign bit.

8.4.9 Additional General-Purpose Registers

The other general-purpose registers are the B, C, D, E, H and L registers and their alternate register set, B', C', D', E', H' and L'. The general-purpose registers can be used interchangeably.

In addition, the B and C registers perform special functions in the NSC800 expanded I/O capabilities, particularly block I/O operations. In these functions, the C register can address I/O ports; the B register provides a counter function when used in the register indirect address mode.

When used with the special condition jump instruction (DJNZ) the B register again provides the counter function.

8.4.10 Alternate Configurations

The six 8-bit general purpose registers (B,C,D,E,H,L) will combine to form three 16-bit registers. This occurs by concatenating the B and C registers to form the BC register, the D and E registers form the DE register, and the H and L registers form the HL register.

Having these 16-bit registers allows 16-bit data handling, thereby expanding the number of 16-bit registers available for memory addressing modes. The HL register typically provides the pointer address for use in register indirect addressing of the memory.

The DE register provides a second memory pointer register for the NSC800's powerful block transfer operations. The BC register also provides an assist to the block transfer operations by acting as a byte-counter for these operations.

8.5 ARITHMETIC-LOGIC UNIT (ALU)

The arithmetic, logic and rotate instructions are performed by the ALU. The ALU internally communicates with the registers and data buffer on the 8-bit internal data bus.

8.6 INSTRUCTION REGISTER AND DECODER

During an opcode fetch, the first byte of an instruction is transferred from the data buffer (i.e. its on the internal data bus) to the instruction register. The instruction register feeds the instruction decoder, which gated by timing signals, generates the control signals that read or write data from or to the registers, control the ALU and provide all required external control signals.

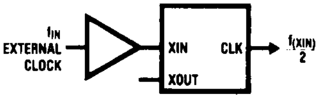
9.0 Timing and Control

9.1 INTERNAL CLOCK GENERATOR

An inverter oscillator contained on the NSC800 chip provides all necessary timing signals. The chip operation frequency is equal to one half of the frequency of this oscillator.

The oscillator frequency can be controlled by one of the following methods:

1. Leaving the X_{OUT} pin unterminated and driving the X_{IN} pin with an externally generated clock as shown in *Figure 6*. When driving X_{IN} with a square wave, the minimum duty cycle is 30% high.

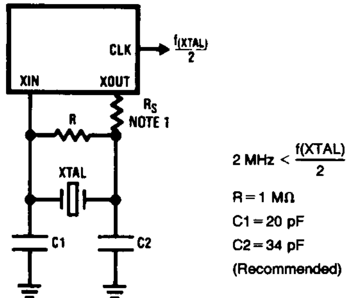


TL/C/5171–13

FIGURE 6. Use of External Clock

2. Connecting a crystal with the proper biasing network between X_{IN} and X_{OUT} as shown in *Figure 7*. Recommended crystal is a parallel resonance AT cut crystal.

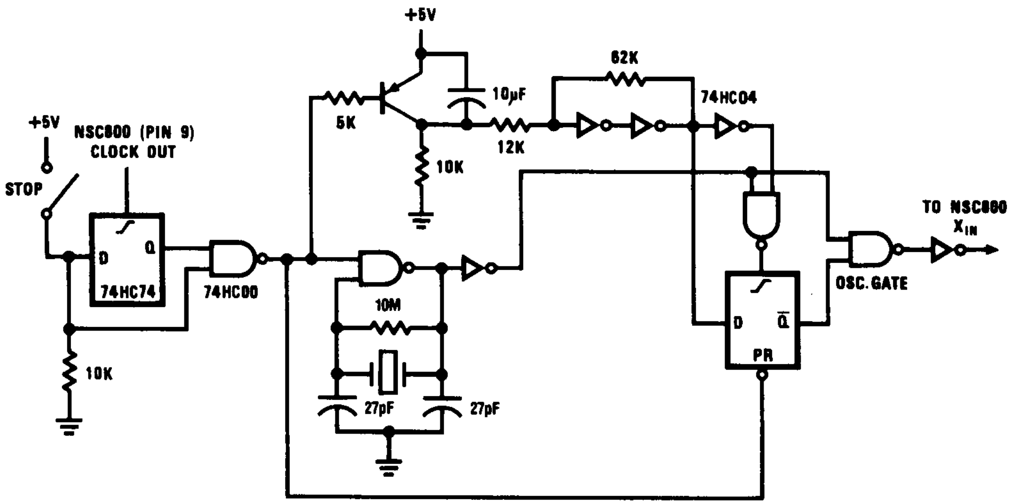
Note 1: If the crystal frequency is between 1 MHz and 2 MHz a series resistor, R_S, (470Ω to 1500Ω) should be connected between X_{OUT} and R, XTAL and C_2. Additionally, the capacitance of C1 and C2 should be increased by 2 to 3 times the recommended value. For crystal frequencies less than 1 MHz higher values of C1 and C2 may be required. Crystal parameters will also affect the capacitive loading requirements.



TL/C/5171–14

FIGURE 7. Use Of Crystal

The CPU has a minimum clock frequency input (@ X_{IN}) of 300 kHz, which results in 150 kHz system clock speed. All registers internal to the chip are static, however there is dynamic logic which limits the minimum clock speed. The input clock can be stopped without fear of losing any data or damaging the part. You stop it in the phase of the clock that has X_{IN} low and CLK OUT high. When restarting the CPU, precautions must be taken so that the input clock meets these minimum specification. Once started, the CPU will continue operation from the same location at which it was stopped. During DC operation of the CPU, typical current drain will be 2 mA. This current drain can be reduced by placing the CPU in a wait state during an opcode fetch cycle then stopping the clock. For clock stop circuit, see *Figure 8*.



TL/C/5171–36

FIGURE 8. Clock Stop Circuit

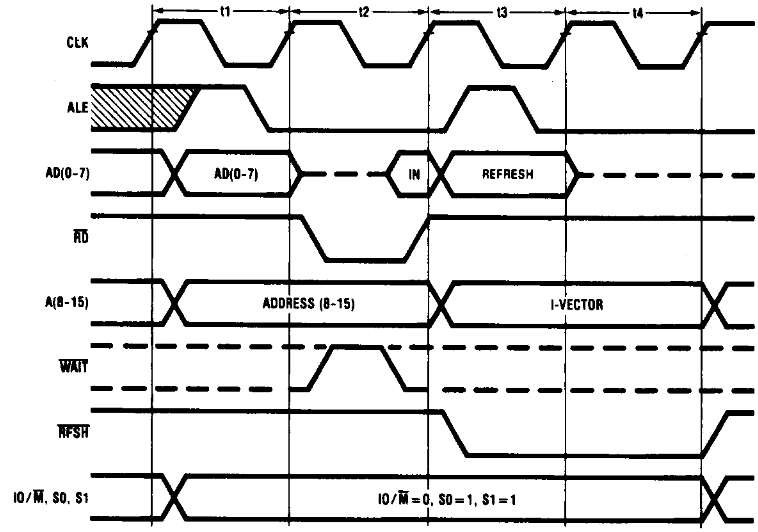
9.0 Timing and Control (Continued)

9.2 CPU TIMING

The NSC800 uses a multiplexed bus for data and addresses. The 16-bit address bus is divided into a high-order 8-bit address bus that handles bits 8–15 of the address, and a low-order 8-bit multiplexed address/data bus that handles bits 0–7 of the address and bits 0–7 of the data. Strobe outputs from the NSC800 (ALE, $\overline{RD}$ and $\overline{WR}$) indicate when a valid address or data is present on the bus. IO/$\overline{M}$ indicates whether the ensuing cycle accesses memory or I/O.

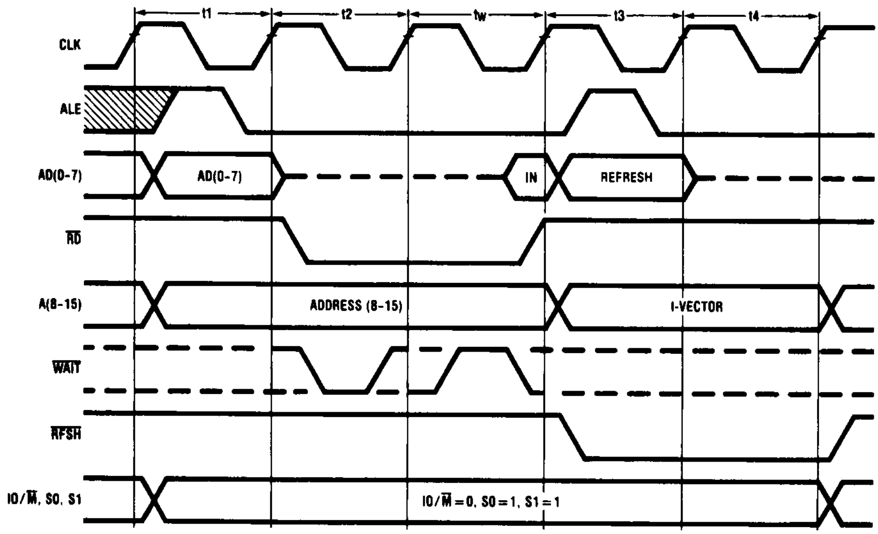
During an input or output instruction, the CPU duplicates the lower half of the address [AD(0–7)] onto the upper address bus [A(8–15)]. The eight bits of address will stay on A(8–15) for the entire machine cycle and can be used for chip selection directly.

Figure 9 illustrates the timing relationship for opcode fetch cycles with and without a wait state.



TL/C/5171-15

FIGURE 9a. Opcode Fetch Cycles without $\overline{WAIT}$ States

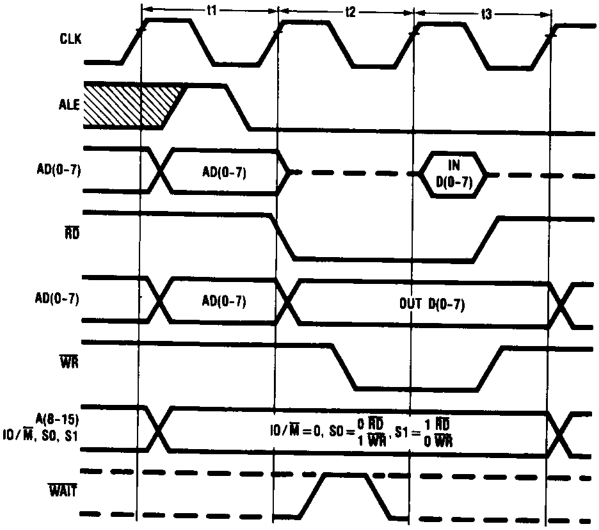


TL/C/5171-16

FIGURE 9b. Opcode Fetch Cycles with $\overline{WAIT}$ States

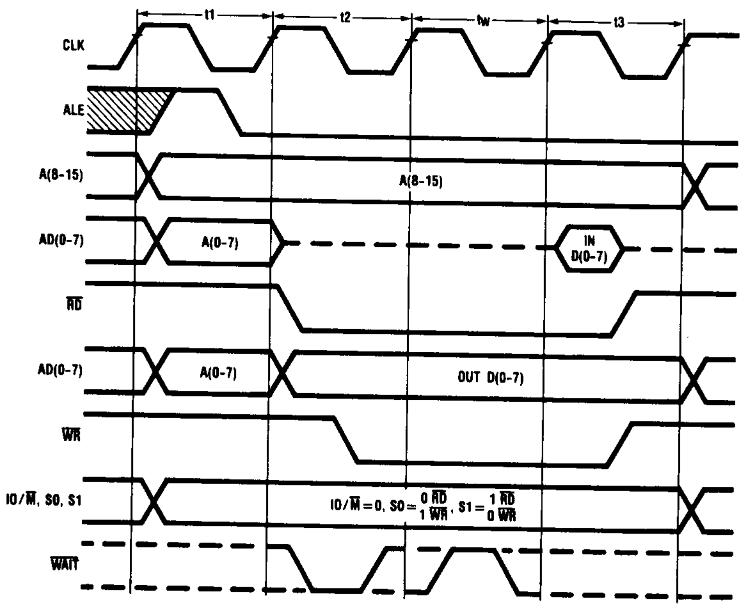
9.0 Timing and Control (Continued)

During the opcode fetch, the CPU places the contents of the PC on the address bus. The falling edge of ALE indicates a valid address on the AD(0–7) lines. The $\overline{\text{WAIT}}$ input is sampled during t_2 and if active causes the NSC800 to insert a wait state (t_w). WAIT is sampled again during t_w so that when it goes inactive, the CPU continues its opcode fetch by latching in the data on the rising edge of $\overline{\text{RD}}$ from the AD(0–7) lines. During t_3, $\overline{\text{RFSH}}$ goes active and AD(0–7) has the dynamic RAM refresh address from register R and A(8–15) the interrupt vector from register I.



TL/C/5171-17

FIGURE 10a. Memory Read/Write Cycles without $\overline{\text{WAIT}}$ States



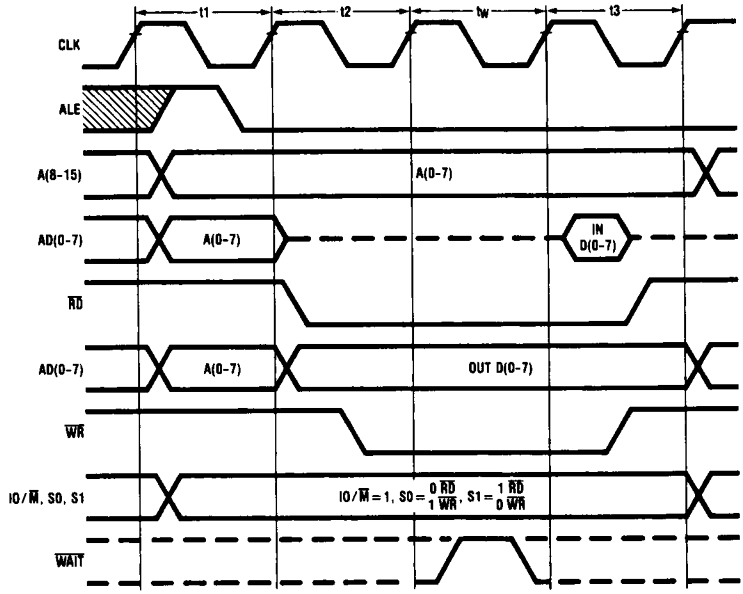
TL/C/5171-18

FIGURE 10b. Memory Read and Write with $\overline{\text{WAIT}}$ States

9.0 Timing and Control (Continued)

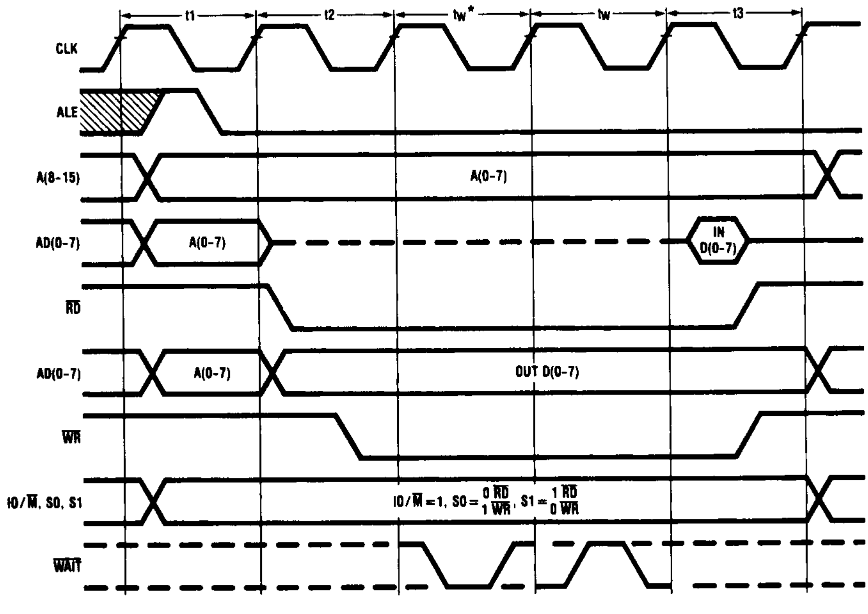
Figure 10 shows the timing for memory read (other than opcode fetchs) and write cycles with and without a wait state. The $\overline{RD}$ stobe is widened by $\frac{t}{2}$ (half the machine state) for memory reads so that the actual latching of the input data occurs later.

Figure 11 shows the timing for input and output cycles with and without wait states. The CPU automatically inserts one wait state into each I/O instruction to allow sufficient time for an I/O port to decode the address.



TL/C/5171-19

FIGURE 11a. Input and Output Cycles without $\overline{WAIT}$ States



TL/C/5171-20

*$\overline{WAIT}$ state automatically inserted during IO operation.

FIGURE 11b. Input and Output Cycles with $\overline{WAIT}$ States

9.0 Timing and Control (Continued)

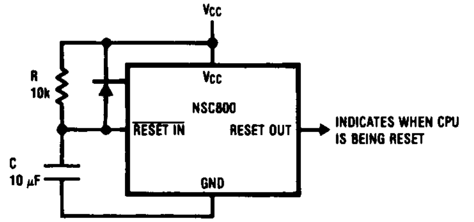
9.3 INITIALIZATION

RESET IN initializes the NSC800; RESET OUT initializes the peripheral components. The Schmitt trigger at the RESET IN input facilitates using an R-C network reset scheme during power up (see *Figure 12*).

To ensure proper power-up conditions for the NSC800, the following power-up and initialization procedure is recommended:

1. Apply power (V_{CC} and GND) and set RESET IN active (low). Allow sufficient time (approximately 30 ms if a crystal is used) for the oscillator and internal clocks to stabilize. RESET IN must remain low for at least 3t state (CLK) times. RESET OUT goes high as soon as the active RESET IN signal is clocked into the first flip-flop after the on-chip Schmitt trigger. RESET OUT signal is available to reset the peripherals.
2. Set RESET IN high. RESET OUT then goes low as the inactive RESET IN signal is clocked into the first flip-flop after the on-chip Schmitt trigger. Following this the CPU initiates the first opcode fetch cycle.

Note: The NSC800 initialization includes: Clear PC to X'0000 (the first opcode fetch, therefore, is from memory location X'0000). Clear registers I (Interrupt Vector Base) and R (Refresh Counter) to X'00. Clear interrupt control register bits IEA, IEB and IEC. The interrupt control bit IEI is set to 1 to maintain INS8080A/Z80A compatibility (see INTERRUPTS for more details). The CPU disables maskable interrupts and enters INTR Mode 0. While RESET IN is active (low), the A(8–15) and AD(0–7) lines go to high impedance (TRI-STATE) and all CPU strobes go to the inactive state (see *Figure 13*).

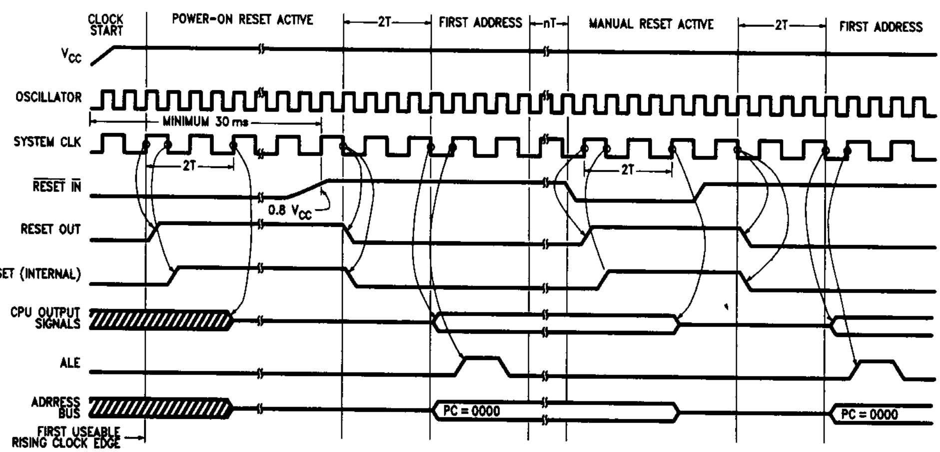


TL/C/5171–21

FIGURE 12. Power-On Reset

9.4 POWER-SAVE FEATURE

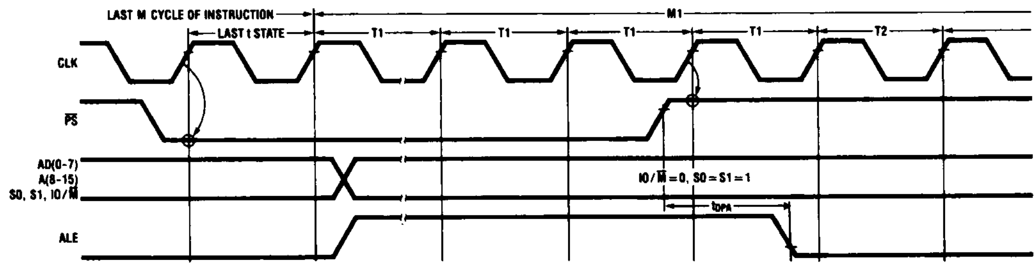
The NSC800 provides a unique power-save mode by the means of the PS pin. PS input is sampled at the last t state of the last M cycle of an instruction. After recognizing an active (low) level on PS, The NSC800 stops its internal clocks, thereby reducing its power dissipation to one half of operating power, yet maintaining all register values and internal control status. The NSC800 keeps its oscillator running, and makes the CLK signal available to the system. When in power-save the ALE strobe will be stopped high and the address lines [AD(0–7), A(8–15)] will indicate the next machine address. When PS returns high, the opcode fetch (or M1 cycle) of the CPU begins in a normal manner. Note this M1 cycle could also be an interrupt acknowledge cycle if the NSC800 was interrupted simultaneously with PS (i.e. PS has priority over a simultaneously occurring interrupt). However, interrupts are not accepted during power save. *Figure 14* illustrates the power save timing.



TL/C/5171–74

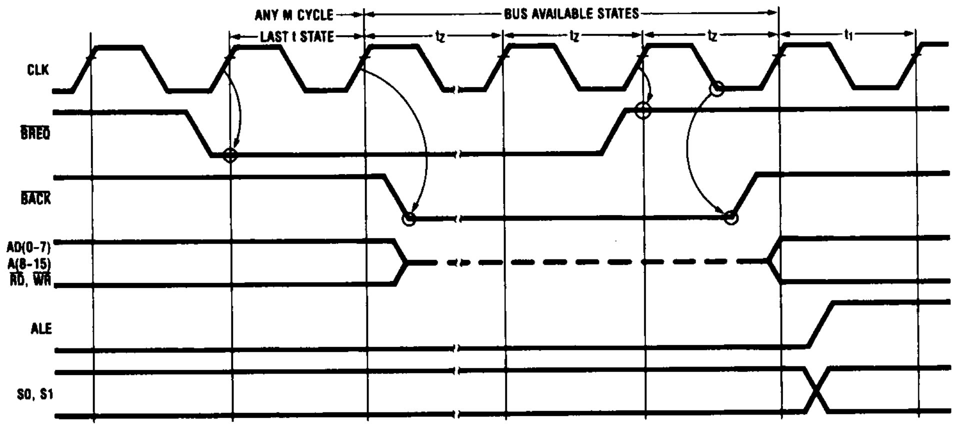
FIGURE 13. NSC800 Signals During Power-On and Manual Reset

9.0 Timing and Control (Continued)



TL/C/5171-28

FIGURE 14. NSC800 Power-Save



TL/C/5171-22

*S0, S1 during $\overline{\text{BREQ}}$ will indicate same machine cycle as during the cycle when $\overline{\text{BREQ}}$ was accepted.

t_Z = time states during which bus and control signals are in high impedance mode.

FIGURE 15. Bus Acknowledge Cycle

In the event $\overline{\text{BREQ}}$ is asserted (low) at the end of an instruction cycle and $\overline{\text{PS}}$ is active simultaneously, the following occurs:

1. The NSC800 will go into $\overline{\text{BACK}}$ cycle.
2. Upon completion of $\overline{\text{BACK}}$ cycle if $\overline{\text{PS}}$ is still active the CPU will go into power-save mode.

9.5 BUS ACCESS CONTROL

Figure 15 illustrates bus access control in the NSC800. The external device controller produces an active $\overline{\text{BREQ}}$ signal that requests the bus. When the CPU responds with $\overline{\text{BACK}}$ then the bus and related control strobes go to high impedance (TRI-STATE) and the $\overline{\text{RFSH}}$ signal remains high. It should be noted that (1) $\overline{\text{BREQ}}$ is sampled at the last t state of any M machine cycle only. (2) The NSC800 will not acknowledge any interrupt/restart requests, and will not peform any dynamic RAM refresh functions until after $\overline{\text{BREQ}}$ input signal is inactive high. (3) $\overline{\text{BREQ}}$ signal has priority over all interrupt request signals, should $\overline{\text{BREQ}}$ and interrupt request become active simultaneously. Therefore, interrupts latched at the end of the instruction cycle will be serviced after a simultaneously occurring $\overline{\text{BREQ}}$. $\overline{\text{NMI}}$ is latched during an active $\overline{\text{BREQ}}$.

9.6 INTERRUPT CONTROL

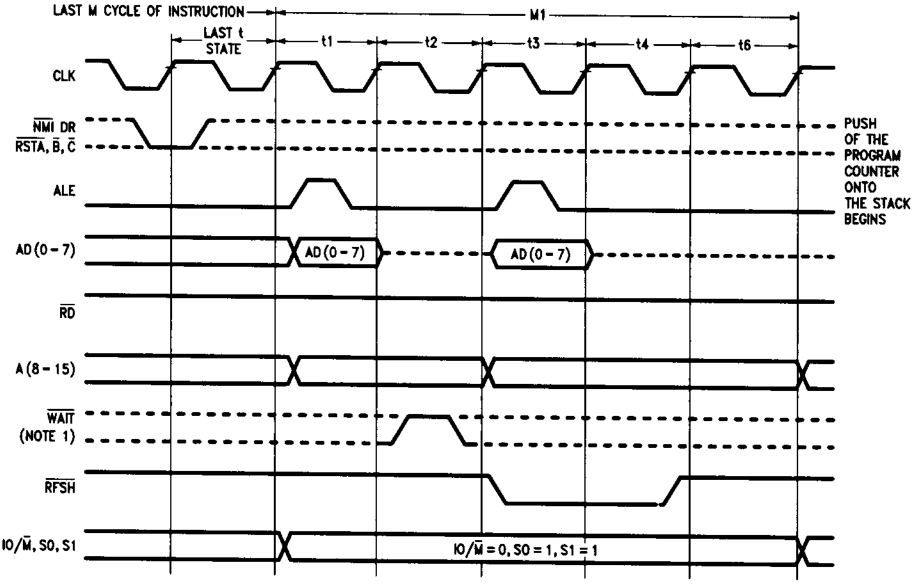
The NSC800 has five interrupt/restart inputs, four are maskable ($\overline{\text{RSTA}}$, $\overline{\text{RSTB}}$, $\overline{\text{RSTC}}$, and $\overline{\text{INTR}}$) and one is non-maskable ($\overline{\text{NMI}}$). $\overline{\text{NMI}}$ has the highest priority of all interrupts; the user cannot disable $\overline{\text{NMI}}$. After recognizing an active input on $\overline{\text{NMI}}$, the CPU stops before the next instruction, pushes the PC onto the stack, and jumps to address X'0066, where the user's interrupt service routine is located (i.e., restart to memory location X'0066). $\overline{\text{NMI}}$ is intended for interrupts requiring immediate attention, such as power-down, control panel, etc.

$\overline{\text{RSTA}}$, $\overline{\text{RSTB}}$ and $\overline{\text{RSTC}}$ are restart inputs, which, if enabled, execute a restart to memory location X'003C, X'0034, and X'002C, respectively. Note that the CPU response to the $\overline{\text{NMI}}$ and $\overline{\text{RST}}$ ($\overline{\text{A}}$, $\overline{\text{B}}$, $\overline{\text{C}}$) request input is basically identical, except for the restored memory location. Unlike $\overline{\text{NMI}}$, however, restart request inputs must be enabled.

Figure 16 illustrates $\overline{\text{NMI}}$ and $\overline{\text{RST}}$ interrupt machine cycles. M1 cycle will be a dummy opcode fetch cycle followed by M2 and M3 which are stack push operations. The following instruction then starts from the interrupts restart location.

Note: $\overline{\text{RD}}$ does *not* go low during this dummy opcode fetch. A unique indication of INTA can be decoded using 2 ALEs and $\overline{\text{RD}}$.

9.0 Timing and Control (Continued)



TL/C/5171-24

Note 1: This is the only machine cycle that does not have an RD, WR, or INTA strobe but will accept a wait strobe.

FIGURE 16. Non-Maskable and Restart Interrupt Machine Cycle

The NSC800 also provides one more general purpose interrupt request input, INTR. When enabled, the CPU responds to INTR in one of the three modes defined by instruction IM0, IM1, and IM2 for modes 0, 1, and 2, respectively. Following reset, the CPU automatically enables mode 0.

Interrupt (INTR) Mode 0: The CPU responds to an interrupt request by providing an INTA (interrupt acknowledge) strobe, which can be used to gate an instruction from a peripheral onto the data bus. The CPU inserts two wait states during the first INTA cycle to allow the interrupting device (or its controller) ample time to gate the instruction and determine external priorities (*Figure 18*). This can be any instruction from one to four bytes. The most popular instruction is one-byte call (restart instruction) or a three-byte call (CALL NN instruction). If it is a three-byte call, the CPU issues a total of three INTA strobes. The last two (which do not include wait states) read NN.

Note: If the instruction stored in the ICU doesn't require the PC to be pushed onto the stack (eq. JP nn), then the PC will not be pushed.

Interrupt (INTR) Mode 1: Similar to restart interrupts except the restart location is X'0038 (*Figure 18*).

Interrupt (INTR) Mode 2: With this mode, the programmer maintains a table that contains the 16-bit starting address of every interrupt service routine. This table can be located anywhere in memory. When the CPU accepts a Mode 2 interrupt (*Figure 17*), it forms a 16-bit pointer to obtain the desired interrupt service routine starting address from the table. The upper 8 bits of this pointer are from the contents of the I register. The lower 8 bits of the pointer are supplied by the interrupting device with the LSB forced to zero. The programmer must load the interrupt vector prior to the interrupt occurring. The CPU uses the pointer to get the two adjacent bytes from the interrupt service routine starting address table to complete 16-bit service routine starting address. The first byte of each entry in the table is the least significant (low-order) portion of the address. The programmer must obviously fill this table with the desired addresses before any interrupts are to be accepted.

Note that the programmer can change this table at any time to allow peripherals to be serviced by different service routines. Once the interrupting device supplies the lower portion of the pointer, the CPU automatically pushes the program counter onto the stack, obtains the starting address from the table and does a jump to this address.

The interrupts have fixed priorities built into the NSC800 as:

NMI	0066	(Highest Priority)
RSTA	003C	
RSTB	0034	
RSTC	002C	
INTR	0038	(Lowest Priority)

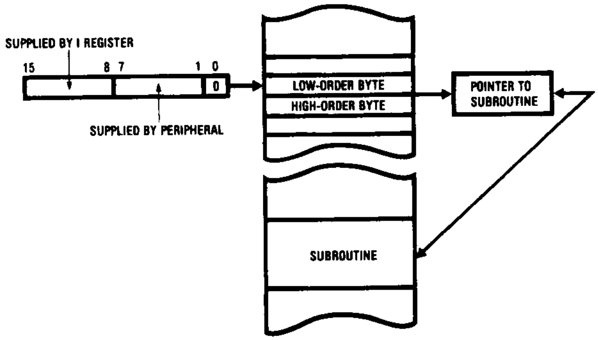
Interrupt Enable, Interrupt Disable. The NSC800 has two types of interrupt inputs, a non-maskable interrupt and four software maskable interrupts. The non-maskable interrupt (NMI) cannot be disabled by the programmer and will be accepted whenever a peripheral device requests an interrupt. The NMI is usually reserved for important functions that must be serviced when they occur, such as imminent power failure. The programmer can selectively enable or disable maskable interrupts (INT, RSTA, RSTB and RSTC). This selectivity allows the programmer to disable the maskable interrupts during periods when timing constraints don't allow program interruption.

There are two interrupt enable flip-flops (IFF_1 and IFF_2) on the NSC800. Two instructions control these flip-flops. Enable Interrupt (EI) and Disable Interrupt (DI). The state of IFF_1 determines the enabling or disabling of the maskable interrupts, while IFF_2 is used as a temporary storage location for the state of IFF_1.

9.0 Timing and Control (Continued)

A reset to the CPU will force both IFF_1 and IFF_2 to the reset state disabling maskable interrupts. They can be enabled by an EI instruction at any time by the programmer. When an EI instruction is executed, any pending interrupt requests will not be accepted until after the instruction following EI has been executed. This single instruction delay is necessary in situations where the following instruction is a return instruction and interrupts must not be allowed until the return has been completed. The EI instruction sets both IFF_1 and IFF_2 to the enable state. When the CPU accepts an interrupt, both IFF_1 and IFF_2 are automatically reset, inhibiting further interrupts until the programmer wishes to issue a new EI instruction. Note that for all the previous cases, IFF_1 and IFF_2 are always equal.

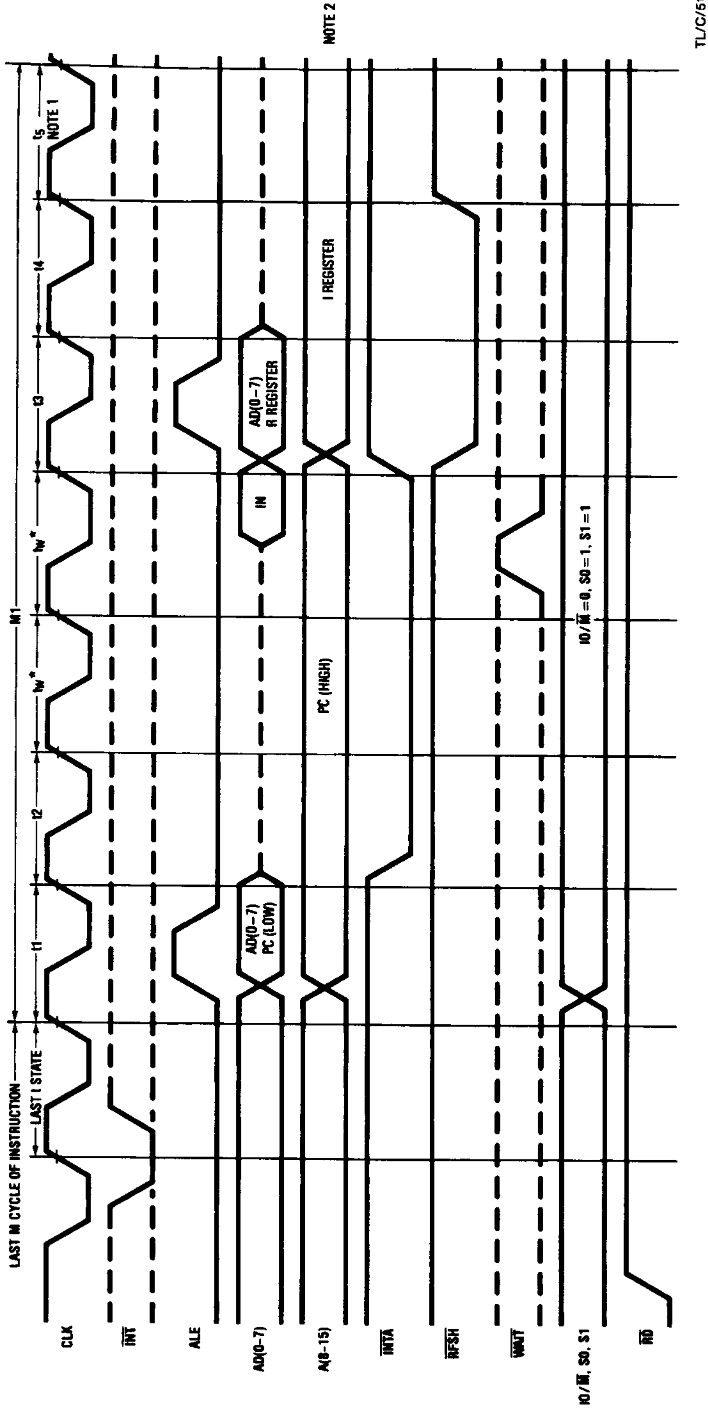
The function of IFF_2 is to retain the status of IFF_1 when a non-maskable interrupt occurs. When a non-maskable interrupt is accepted, IFF_1 is reset to prevent further interrupts until reenabled by the programmer. Thus, after a non-maskable interrupt has been accepted, maskable interrupts are disabled but the previous state of IFF_1 is saved by IFF_2



TL/C/5171-27

FIGURE 17. Interrupt Mode 2

9.0 Timing and Control (Continued)



TL/C/5171-25

*tw is the CPU generated WAIT state in response to an interrupt request.

Note 1: t5 will only occur in mode 1 and mode 2. During t5 the stack pointer is decremented.

Note 2: A jump to the appropriate address occurs here in mode 1 and mode 2. The CPU continues gathering data from the interrupting peripheral in mode 0 for a total of 2–4 machine cycles. In mode 0 cycles M2–M4 have only 1 wait state.

FIGURE 18. Interrupt Acknowledge Machine Cycle

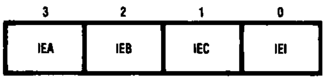
9.0 Timing and Control (Continued)

so that the complete state of the CPU just prior to the non-maskable interrupt may be restored. The method of restoring the status of IFF_1 is through the execution of a Return Non-Maskable Interrupt (RETN) instruction. Since this instruction indicates that the non-maskable interrupt service routine is completed, the contents of IFF_2 are now copied back into IFF_1, so that the status of IFF_1 just prior to the acceptance of the non-maskable interrupt will be automatically restored.

Figure 19 depicts the status of the flip flops during a sample series of interrupt instructions.

Interrupt Control Register. The interrupt control register (ICR) is a 4-bit, write only register that provides the programmer with a second level of maskable control over the four maskable interrupt inputs.

The ICR is internal to the NSC800 CPU, but is addressed through the I/O space at I/O address port X'BB. Each bit in the register controls a mask bit dedicated to each maskable interrupt, $\overline{RSTA}$, $\overline{RSTB}$, $\overline{RSTC}$ and $\overline{INTR}$. For an interrupt request to be accepted on any of these inputs, the corresponding mask bit in the ICR must be set (= 1) and IFF_1 and IFF_2 must be set. This provides the programmer with control over individual interrupt inputs rather than just a system wide enable or disable.



TL/C/5171-26

Bit	Name	Function
0	IEI	Interrupt Enable for $\overline{INTR}$
1	IEC	Interrupt Enable for $\overline{RSTC}$
2	IEB	Interrupt Enable for $\overline{RSTB}$
3	IEA	Interrupt Enable for $\overline{RSTA}$

For example: In order to enable $\overline{RSTB}$, CPU interrupts must be enabled and IEB must be set.

At reset, IEI bit is set and other mask bits IEA, IEB, IEC are cleared. This maintains the software compatibility between NSC800 and Z80A.

Execution of an I/O block move instruction will not affect the state of the interrupt control bits. The only two instructions that will modify this write only register are OUT (C), r and OUT (N), A.

Operation	IFF_1	IFF_2	Comment
Initialize	0	0	Interrupt Disabled
•			
•			
•			
EI	1	1	Interrupt Enabled after next instruction
•			
•			
•			
$\overline{INTR}$	0	0	Interrupt Disable and $\overline{INTR}$ Being Serviced
•			
•			
•			
EI	1	1	Interrupt Enabled after next instruction
RET	1	1	Interrupt Enabled
•			
•			
•			
$\overline{NMI}$	0	1	Interrupt Disabled
•			
•			
•			
RETN	1	1	Interrupt Enabled
•			
$\overline{INTR}$	0	0	Interrupt Disabled
•			
•			
•			
$\overline{NMI}$	0	0	Interrupt Disabled and $\overline{NMI}$ Being Serviced
•			
•			
•			
RETN	0	0	Interrupt Disabled and $\overline{INTR}$ Being Serviced
•			
•			
•			
EI	1	1	Interrupt Enabled after next instruction
RET	1	1	Interrupt Enabled
•			
•			
•			

FIGURE 19. IFF_1 and IFF_2 States Immediately after the Operation has been Completed

NSC800 SOFTWARE

10.0 Introduction

This chapter provides the reader with a detailed description of the NSC800 software. Each NSC800 instruction is described in terms of opcode, function, flags affected, timing, and addressing mode.

11.0 Addressing Modes

The following sections describe the addressing modes supported by the NSC800. Note that particular addressing modes are often restricted to certain types of instructions. Examples of instructions used in the particular addressing modes follow each mode description.

The 10 addressing modes and 158 instructions provide a flexible and powerful instruction set.

11.1 REGISTER

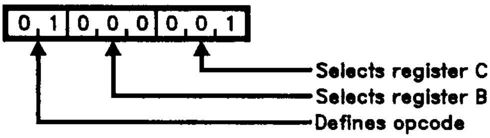
The most basic addressing mode is that which addresses data in the various CPU registers. In these cases, bits in the opcode select specific registers that are to be addressed by the instruction.

Example:

Instruction: Load register B from register C

Mnemonic: LD B,C

Opcode:



TL/C/5171-50

In this instruction, both the B and C registers are addressed by opcode bits.

11.2 IMPLIED

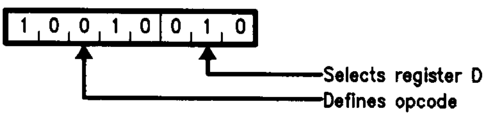
The implied addressing mode is an extension to the register addressing mode. In this mode, a specific register, the accumulator, is used in the execution of the instruction. In particular, arithmetic operations employ implied addressing, since the A register is assumed to be the destination register for the result without being specifically referenced in the opcode.

Example:

Instruction: Subtract the contents of register D from the Accumulator (A register)

Mnemonic: SUB D

Opcode:



TL/C/5171-51

In this instruction, the D register is addressed with register addressing, while the use of the A register is implied by the opcode.

11.3 IMMEDIATE

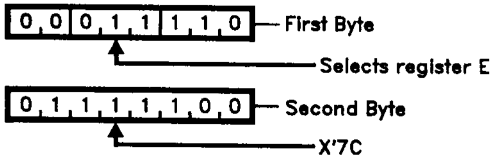
The most straightforward way of introducing data to the CPU registers is via immediate addressing, where the data is contained in an additional byte of multi-byte instructions.

Example:

Instruction: Load the E register with the constant value X'7C.

Mnemonic: LD E,X'7C

Opcode:



TL/C/5171-52

In this instruction, the E register is addressed with register addressing, while the constant X'7C is immediate data in the second byte of the instruction.

11.4 IMMEDIATE EXTENDED

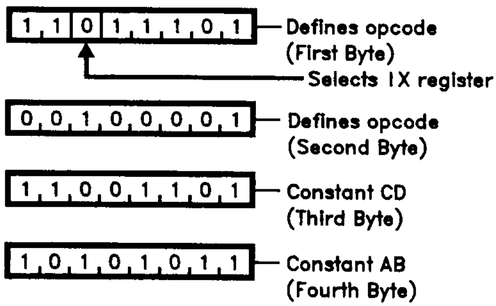
As immediate addressing allows 8 bits of data to be supplied by the operand, immediate extended addressing allows 16 bits of data to be supplied by the operand. These are in two additional bytes of the instruction.

Example:

Instruction: Load the 16-bit IX register with the constant value X'ABCD.

Mnemonic: LD IX,X'ABCD

Opcode:



TL/C/5171-53

In this instruction, register addressing selects the IX register, while the 16-bit quanity X'ABCD is immediate data supplied as immediate extended format.

11.0 Addressing Modes (Continued)

11.5 DIRECT ADDRESSING

Direct addressing is the most straightforward way of addressing supplies a location in the memory space. Direct addressing, 16-bits of memory address information in two bytes of data as part of the instruction. The memory address could be either data, source of destination, or a location for program execution, as in program control instructions.

Example:

Instruction: Jump to location X'0377

Mnemonic: JP X'0377

Opcode:

1	1	0	0	0	0	1	1	—Defines jump opcode
0	1	1	1	0	1	1	1	—Constant X'0377
0	0	0	0	0	0	1	1	

This instruction loads the Program Counter (PC) is loaded with the constant in the second and third bytes of the instruction. The program counter contents are transferred via direct addressing.

11.6 REGISTER INDIRECT

Next to direct addressing, register indirect addressing provides the second most straightforward means of addressing memory. In register indirect addressing, a specified register pair contains the address of the desired memory location. The instruction references the register pair and the register contents define the memory location of the operand.

Example:

Instruction: Add the contents of memory location X'0254 to the A register. The HL register contains X'0254.

Mnemonic: ADD A,(HL)

Opcode

1	0	0	0	0	1	1	0

This instruction uses implied addressing of the A and HL registers and register indirect addressing to access the data pointed to by the HL register.

11.7 INDEXED

The most flexible mode of memory addressing is the indexed mode. This is similar to the register indirect mode of addressing because one of the two index registers (IX or IY) contains the base memory address. In addition, a byte of data included in the instruction acts as a displacement to the address in the index register.

Indexed addressing is particularly useful in dealing with lists of data.

Example:

Instruction: Increment the data in memory location X'1020. The IY register contains X'1000.

Mnemonic: INC (IY + X'20)

Opcode:

1	1	1 (Selects IY register)	1	1	1	0	1	Defines increment opcode
0	0	1	1	0	1	0	0	
0	0	1	0	0	0	0	0	Displacement to IY index register (Third Byte)

TL/C/5171-54

The indexed addressing mode uses the contents of index registers IX or IY along with the displacement to form a pointer to memory.

11.8 RELATIVE

Certain instructions allow memory locations to be addressed as a position relative to the PC register. These instructions allow jumps to memory locations which are offsets around the program counter. The offset, together with the current program location, is determined through a displacement byte included in the instruction. The formation of this displacement byte is explained more fully in the "Instructions Set" section.

Example:

Instruction: Jump to a memory location 7 bytes beyond the current location.

Mnemonic: JR \$+7

Opcode:

0	0	0	1	1	0	0	0	—Defines relative jump opcode
0	0	0	0	0	1	0	1	—Displacement to be applied to the PC

The program will continue at a location seven locations past the current PC.

11.0 Addressing Modes (Continued)

11.9 MODIFIED PAGE ZERO

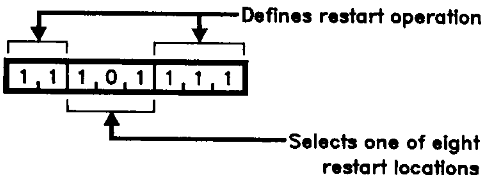
A subset of NSC800 instructions (the Restart instructions) provides a code-efficient single-byte instruction that allows CALLs to be performed to any one of eight dedicated locations in page zero (locations X'0000 to X'00FF). Normally, a CALL is a 3-byte instruction employing direct memory addressing.

Example:

Instruction: Perform a restart call to location X'0028.

Mnemonic: RST X'28

Opcode:



TL/C/5171-55

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

Program execution continues at location X'0028 after execution of a single-byte call employing modified page zero addressing.

11.10 BIT

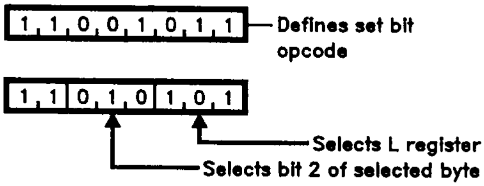
The NSC800 allows setting, resetting, and testing of individual bits in registers and memory data bytes.

Example:

Operation: Set bit 2 in the L register

Mnemonic: SET 2,L

Opcode:



TL/C/5171-56

Bit addressing allows the selection of bit 2 in the L register selected by register addressing.

12.0 Instruction Set

This section details the entire NSC800 instruction set in terms of

- Opcode
- Instruction
- Function
- Timing
- Addressing Mode

The instructions are grouped in order under the following functional headings:

- 8-Bit Loads
- 16-Bit Loads
- 8-Bit Arithmetic
- 16-Bit Arithmetic
- Bit Set, Reset, and Test
- Rotate and Shift
- Exchanges
- Memory Block Moves and Searches
- Input/Output
- CPU Control
- Program Control

12.1 Instruction Set Index

Alphabetical Assembly Mnemonic	Operation
ADC A,m_1	Add, with carry, memory location contents to Accumulator
ADC A,n	Add, with carry, immediate data n to Accumulator
ADC A,r	Add, with carry, register r contents to Accumulator
ADC HL,pp	Add, with carry, register pair pp to HL
ADD A,m_1	Add memory location contents to Accumulator
ADD A,n	Add immediate data n to Accumulator
ADD A,r	Add register r contents to Accumulator
ADD HL,pp	Add register pair pp to HL
ADD IX,pp	Add register pair pp to IX
ADD IY,pp	Add register pair pp to IY
ADD ss,pp	Add register pair pp to contents of register pair ss
AND m_1	Logical 'AND' memory contents to Accumulator
AND n	Logical 'AND' immediate data to Accumulator
AND r	Logical 'AND' register r contents to Accumulator
BIT b,m_1	Test bit b of location m_1
BIT b,r	Test bit b of register r
CALL cc,nn	Call subroutine at location nn if condition cc is true
CALL nn	Unconditional call to subroutine at location nn
CCF	Complement carry flag
CP m_1	Compare memory contents with Accumulator
CP n	Compare immediate data n with Accumulator
CP r	Compare register r to contents with Accumulator
CPD	Compare location (HL) and Accumulator, decrement HL and BC
CPDR	Compare location (HL) and Accumulator, decrement HL and BC; repeat until BC = 0
CPI	Compare location (HL) and Accumulator, increment HL, decrement BC
CPIR	Compare location (HL) and Accumulator, increment HL, decrement BC; repeat until BC = 0
CPL	Complement Accumulator (1's complement)
DAA	Decimal adjust Accumulator
DEC m_1	Decrement data in memory location m_1
DEC r	Decrement register r contents
DEC rr	Decrement register pair rr contents

12.1 Instruction Set Index (Continued)

Alphabetical Assembly Mnemonic	Operation
DI	Disable interrupts
DJNZ,d	Decrement B and jump relative $B \neq 0$
EI	Enable interrupts
EX (SP),ss	Exchange the location (SP) with register ss
EX AF,A'F'	Exchange the contents of AF and A'F'
EX DE,HL	Exchange the contents of DE and HL
EXX	Exchange the contents of BC, DE and HL with the contents of B'C, D'E' and H'L', respectively
HALT	Halt (wait for interrupt or reset)
IM 0	Set interrupt mode 0
IM 1	Set interrupt mode 1
IM 2	Set interrupt mode 2
IN A,(n)	Load Accumulator with input from device (n)
IN r,(C)	Load register r with input from device (C)
INC m_1	Increment data in memory location m_1
INC r	Increment register r
INC rr	Increment contents of register pair rr
IND	Load location (HL) with input from port (C), decrement HL and B
INDR	Load location (HL) with input from port (C), decrement HL and B; repeat until B = 0
INI	Load location (HL) with input from port (C), increment HL, decrement B
INIR	Load location (HL) with input from port (C), increment HL, decrement B; repeat until B = 0
JP cc,nn	Jump to location nn, if condition cc is true
JP nn	Unconditional jump to location nn
JP (ss)	Unconditional jump to location (ss)
JR d	Unconditional jump relative to PC + d
JR kk,d	Jump relative to PC + d, if kk true
LD A,I	Load Accumulator with register I contents
LD A,m_2	Load Accumulator from location m_2
LD A,R	Load Accumulator with register R contents
LD I,A	Load register I with Accumulator contents
LD m_1,n	Load memory with immediate data n
LD m_1,r	Load memory from register r
LD m_2,A	Load memory from Accumulator
LD (nn),rr	Load memory location nn with register pair rr
LD r,m_1	Load register r from memory
LD r,n	Load register with immediate data n
LD R,A	Load register R from Accumulator
LD r_d,r_s	Load destination register r_d from source register r_s
LD rr,(nn)	Load register pair rr from memory location nn
LD rr,nn	Load register pair rr with immediate data nn
LD SP,ss	Load SP from register pair ss
LDD	Load location (DE) with location (HL), decrement DE, HL and BC
LDDR	Load location (DE) with location (HL), decrement DE, HL and BC; repeat until BC = 0
LDI	Load location (DE) with location (HL), increment DE and HL, decrement BC
LDIR	Load location (DE) with location (HL), increment DE and HL, decrement BC; repeat until BC = 0
NEG	Negate Accumulator (2's complement)
NOP	No operation

12.1 Instruction Set Index (Continued)

Alphabetical Assembly Mnemonic	Operation
OR m_1	Logical 'OR' of memory location contents and accumulator
OR n	Logical 'OR' of immediate data n and Accumulator
OR r	Logical 'OR' of register r and Accumulator
OTDR	Load output port (C) with location (HL), decrement HL and B; repeat until B = 0
OTIR	Load output port (C) with location (HL), increment HL, decrement B; repeat until B = 0
OUT (C),r	Load output port (C) with register r
OUT (n),A	Load output port (n) with Accumulator
OUTD	Load output port (C) with location (HL), decrement HL and B
OUTI	Load output port (C) with location (HL), increment HL, decrement B
POP qq	Load register pair qq with top of stack
PUSH qq	Load top of stack with register pair qq
RES b,m_1	Reset bit b of memory location m_1
RES b,r	Reset bit b of register r
RET	Unconditional return from subroutine
RET cc	Return from subroutine, if cc true
RETI	Unconditional return from interrupt
RETN	Unconditional return from non-maskable interrupt
RL m_1	Rotate memory contents left through carry
RL r	Rotate register r left through carry
RLA	Rotate Accumulator left through carry
RLC m_1	Rotate memory contents left circular
RLC r	Rotate register r left circular
RLCA	Rotate Accumulator left circular
RLD	Rotate digit left and right between Accumulator and memory (HL)
RR m_1	Rotate memory contents right through carry
RR r	Rotate register r right through carry
RRA	Rotate Accumulator right through carry
RRC m_1	Rotate memory contents right circular
RRC r	Rotate register r right circular
RRCA	Rotate Accumulator right circular
RRD	Rotate digit right and left between Accumulator and memory (HL)
RST P	Restart to location P
SBC A,m_1	Subtract, with carry, memory contents from Accumulator
SBC A,n	Subtract, with carry, immediate data n from Accumulator
SBC A,r	Subtract, with carry, register r from Accumulator
SBC HL,pp	Subtract, with carry, register pair pp from HL
SCF	Set carry flag
SET b,m_1	Set bit b in memory location m_1 contents
SET b,r	Set bit b in register r
SLA m_1	Shift memory contents left, arithmetic
SLA r	Shift register r left, arithmetic
SRA m_1	Shift memory contents right, arithmetic
SRA r	Shift register r right, arithmetic
SRL m_1	Shift memory contents right, logical
SRL r	Shift register r right, logical
SUB m_1	Subtract memory contents from Accumulator
SUB n	Subtract immediate data n from Accumulator
SUB r	Subtract register r from Accumulator
XOR m_1	Exclusive 'OR' memory contents and Accumulator
XOR n	Exclusive 'OR' immediate data n and Accumulator
XOR r	Exclusive 'OR' register r and Accumulator

12.0 Instruction Set (Continued)

12.2 INSTRUCTION SET MNEMONIC NOTATION

In the following instruction set listing, the notations used are shown below.

b: Designates one bit in a register or memory location. Bit address mode uses this indicator.

cc: Designates condition codes used in conditional Jumps, Calls, and Return instruction; may be:

NZ = Non-Zero (Z flag = 0)
Z = Zero (Z flag = 1)
NC = Non-Carry (C flag = 0)
C = Carry (C flag = 1)
PO = Parity Odd or No Overflow (P/V = 0)
PE = Parity Even or Overflow (P/V = 1)
P = Positive (S = 0)
M = Negative (S = 1)

d: Designates an 8-bit signed complement displacement. Relative or indexed address modes use this indicator.

kk: Subset of cc condition codes used in conjunction with conditional relative jumps; may be NZ, Z, NC or C.

m_1: Designates (HL), (IX+d) or (IY+d). Register indirect or indexed address modes use this indicator.

m_2: Designates (BC), (DE) or (nn). Register indirect or direct address modes use this indicator.

n: Any 8-bit binary number.

nn: Any 16-bit binary number.

p: Designates restart vectors and may be the hex values 0, 8, 10, 18, 20, 28, 30 or 38. Restart instructions employing the modified page zero addressing mode use this indicator.

pp: Designates the BC, DE, SP or any 16-bit register used as a destination operand in 16-bit arithmetic operations employing the register address mode.

qq: Designates BC, DE, HL, A, F, IX, or IY during operations employing register address mode.

r: Designates A, B, C, D, E, H or L. Register addressing modes use this indicator.

rr: Designates BC, DE, HL, SP, IX or IY. Register addressing modes use this indicator.

ss: Designates HL, IX or IY. Register addressing modes use this indicator.

X_L: Subscript L indicates the lower-order byte of a 16-bit register.

X_H: Subscript H indicates the high-order byte of a 16-bit register.

( ): parentheses indicate the contents are considered a pointer address to a memory or I/O location.

12.3 ASSEMBLED OBJECT CODE NOTATION

Register Codes:

r	Register	rp	Register	rs	Register
000	B	00	BC	00	BC
001	C	01	DE	01	DE
010	D	10	HL	10	HL
011	E	11	SP	11	AF
		pp	**Register**	**qq**	**Register**
100	H	00	BC	00	BC
101	L	01	DE	01	DE
111	A	10	IX	10	HL
		11	SP	11	AF

Conditions Codes:

cc	Mnemonic	True Flag Condition
000	NZ	Z=0
001	Z	Z=1
010	NC	C=0
011	C	C=1
100	PO	P/V=0
101	PE	P/V=1
110	P	S=0
111	M	S=1
kk	**Mnemonic**	**True Flag Condition**
00	NZ	Z=0
01	Z	Z=1
10	NC	C=0
11	C	C=1

Restart Addresses:

t	T
000	X'00
001	X'08
010	X'10
011	X'18
100	X'20
101	X'28
110	X'30
111	X'38

12.4 8-Bit Loads

REGISTER TO REGISTER

LD r_d, r_s

Load register r_d with r_s:

$r_d \leftarrow r_s$ — No flags affected

7	6	5	4	3	2	1	0
0	1		r_d			r_s	

Timing: M cycles — 1
T states — 4

Addressing Mode: Register

LD A, I

Load Accumulator with the contents of the I register.

$A \leftarrow I$

S: Set if negative result
Z: Set if zero result
H: Reset
P/V: Set according to IFF_2 (zero if interrupt occurs during operation)
N: Reset
C: Not affected

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	0	1	0	1	1	1

Timing: M cycles — 2
T states — 9 (4, 5)

Addressing Mode: Register

LD I, A

Load Interrupt vector register (I) with the contents of A.

$I \leftarrow A$ — No flags affected

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	0	0	0	1	1	1

Timing: M cycles — 2
T states — 9 (4, 5)

Addressing Mode: Register

LD A, R

Load Accumulator with contents of R register.

$A \leftarrow R$

S: Set if negative result
Z: Set if zero result
H: Reset
P/V: Set according to IFF_2 (zero if interrupt occurs during operation)
N: Reset
C: Not affected

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	0	1	1	1	1	1

Timing: M cycles — 2
T states — 9 (4, 5)

Addressing Mode: Register

LD R, A

Load Refresh register (R) with contents of the Accumulator.

$R \leftarrow A$ — No flags affected

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	0	0	1	1	1	1

Timing: M cycles — 2
T states — 9 (4, 5)

Addressing Mode: Register

LD r, n

Load register r with immediate data n.

$r \leftarrow n$ — No flags affected

7	6	5	4	3	2	1	0
0	0		r		1	1	0
			n				

Timing: M cycles — 2
T states — 7 (4, 3)

Addressing Mode: Source — Immediate
Destination — Register

REGISTER TO MEMORY

LD m_1, r

Load memory from reigster r.

$m_1 \leftarrow r$ — No flags affected

7	6	5	4	3	2	1	0	
0	1	1	1	0		r		LD (HL), r

Timing: M cycles — 2
T states — 7 (4,3)

Addressing Mode: Source — Register
Destination — Register Indirect

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	LD (IX + d), r(for $N_X = 0$) LD (IY + d), r(for $N_X = 1$)
0	1	1	1	0		r		
			d					

Timing: M cycles — 2
T states — 19 (4, 4, 3, 5, 3)

Addressing Mode: Source — Register
Destination — Indexed

12.4 8-Bit Loads (Continued)

LD m₂, A

Load memory from the Accumulator.

$m_2 \leftarrow A$ — No flags affected

7	6	5	4	3	2	1	0	
0	0	0	0	0	0	1	0	LD (BC), A
0	0	0	1	0	0	1	0	LD (DE), A

Timing: M cycles — 2
T states — 7 (4, 3)

Addressing Mode: Source — Register (Implied)
Destination — Register Indirect

7	6	5	4	3	2	1	0	
0	0	1	1	0	0	1	0	LD (nn), A
n (low-order byte)								
n (high-order byte)								

Timing: M cycles — 4
T states — 3 (4, 3, 3, 3)

Addressing Mode: Source — Register (Implied)
Destination — Direct

LD m₁, n

Load memory with immediate data.

$m_1 \leftarrow n$ — No flags affected

7	6	5	4	3	2	1	0	
0	0	1	1	0	1	1	0	LD(HL), n
n								

Timing: M cycles—3
T states—10 (4, 3, 3)

Addressing Mode: Source—Immediate
Destination—Register Indirect

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	LD (IX + d), n(for N_X = 0) LD (IY + d), n(for N_X = 1)
0	0	1	1	0	1	1	0	
d								
n								

Timing: M cycles—5
T states—19 (4, 4, 3, 5, 3)

Addressing Mode: Source—Immediate
Destination—Indexed

MEMORY TO REGISTER

LD r, m₁

Load register r from memory location m_1.

$r \leftarrow m_1$ — No flags affected

7	6	5	4	3	2	1	0	
0	1	r			1	1	0	LD R, (HL)

Timing: M cycles—2
T states—7 (4, 3)

Addressing Mode: Source—Register Indirect
Destination—Register

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	LD r, (IX + d) (for N_X=0) LD r, (IY + d) (for N_X=1)
0	1	r			1	1	0	
d								

Timing: M cycles—5
T states—19 (4, 4, 3, 5, 3)

Addressing Mode: Source—Indexed
Destination—Register

LD A, m₂

Load the Accumulator from memory location m_2.

$A \leftarrow m_2$ — No flags affected

7	6	5	4	3	2	1	0	
0	0	0	0	1	0	1	0	LD A, (BC)
0	0	0	1	1	0	1	0	LD A, (DE)

Timing: M cycles—2
T states—7 (4, 3)

Addressing Mode: Source—Register Indirect
Destination—Register (Implied)

7	6	5	4	3	2	1	0	
0	0	1	1	1	0	1	0	LD A, (nn)
n (low-order byte)								
n (high-order byte)								

Timing: M cycles—4
T states—13 (4, 3, 3, 3)

Addressing Mode: Source—Immediate Extended
Destination—Register (Implied)

12.5 16-Bit Loads

REGISTER TO REGISTER

LD rr, nn

Load 16-bit register pair with immediate data.

rr, ← nn — No flags affected

7	6	5	4	3	2	1	0
0	0	rp		0	0	0	1
n (low-order byte)							
n (high-order byte)							

LD BC, nn
LD DE, nn
LD HL, nn
LD SP, nn

Timing: M cycles—3
T states—10 (4, 3, 3)

Addressing Mode: Source—Immediate Extended
Destination—Register

7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
0	0	1	0	0	0	0	1
n (low-order byte)							
n (high-order byte)							

LD IX, nn (for $N_X = 0$)
LD IY, nn (for $N_X = 1$)

Timing: M cycles—4
T states—14 (4, 4, 3, 3)

Addressing Mode: Source—Immediate Extended
Destination—Register

LD SP, ss

Load the SP from 16-bit register ss.

SP ← ss — No flags affected

7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	1

LD SP, HL

Timing: M cycles—1
T states—6

Addressing Mode: Source—Register
Destination—Register (Implied)

7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
1	1	1	1	1	0	0	1

LD SP, IX (for $N_X = 0$)
LD SP, IY (for $N_X = 1$)

Timing: M cycles—2
T states—10 (4, 6)

Addressing Mode: Source—Register
Destination—Register (Implied)

REGISTER TO MEMORY

LD (nn), rr

Load memory location nn with contents of 16-bit register, rr.

(nn) ← rr_L — No flags affected

(nn + 1) ← rr_H

7	6	5	4	3	2	1	0
0	0	1	0	0	0	1	0
n (low-order byte)							
n (high-order byte)							

LD (nn), HL
(note an alternate opcode below)

Timing: M cycles—5
T states—16 (4, 3, 3, 3, 3)

Addressing Mode: Source—Register
Destination—Direct

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	rp		0	0	1	1
n (low-order byte)							
n (high-order byte)							

LD (nn), BC
LD (nn), DE
LD (nn), HL
LD (nn), SP

Timing: M cycles—6
T states—20 (4, 4, 3, 3, 3, 3)

Addressing Mode: Source—Register
Destination—Direct

7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
0	0	1	0	0	0	1	0
n (low-order byte)							
n (high-order byte)							

LD (nn), IX (for $N_X = 0$)
LD (nn) IY (for $N_X = 1$)

Timing: M cycles—6
T states—20 (4, 4, 3, 3, 3, 3)

Addressing Mode: Source—Register
Destination—Direct

12.5 16-Bit Loads (Continued)

PUSH qq

Push the contents of register pair qq onto the memory stack.

$(SP - 1) \leftarrow qq_H$ — No flags affected

$(SP - 2) \leftarrow qq_L$

$SP \leftarrow SP - 2$

7	6	5	4	3	2	1	0
1	1	rs		0	1	0	1

PUSH BC
PUSH DE
PUSH HL
PUSH AF

Timing: M cycles—3
T states—11 (5, 3, 3)

Addressing Mode: Source—Register
Destination—Register Indirect (Stack)

7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
1	1	1	0	0	1	0	1

PUSH IX (for $N_X = 0$)
PUSH IY (for $N_X = 1$)

Timing: M cycles—3
T states—15 (4, 5, 3, 3)

Addressing Mode: Source—Register
Destination—Register Indirect (Stack)

MEMORY TO REGISTER

LD rr, (nn)

Load 16-bit register from memory location nn.

$rr_L \leftarrow (nn)$ — No flags affected

$rr_H \leftarrow (nn + 1)$

7	6	5	4	3	2	1	0
0	0	1	0	1	0	1	0
n (low-order byte)							
n (high-order byte)							

LD HL, (nn)
(note an alternate opcode below)

Timing: M cycles—5
T states—16 (4, 3, 3, 3, 3)

Addressing Mode: Source—Direct
Destination—Register

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	rp		0	0	1	1
n (low-order byte)							
n (high-order byte)							

LD BC, (nn)
LD DE, (nn)
LD HL, (nn)
LD SP, (nn)

Timing: M cycles—6
T states—20 (4, 4, 3, 3, 3, 3)

Addressing Mode: Source—Direct
Destination—Register

7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
0	0	1	0	1	0	1	0
n (low-order byte)							
n (high-order byte)							

LD IX, (nn)(for $N_X = 0$)
LD IY, (nn) (for $N_X = 1$)

Timing: M cycles—6
T states—20 (4, 4, 3, 3, 3, 3)

Addressing Mode: Source—Direct
Destination—Register

POP qq

Pop the contents of the memory stack to register qq.

$qq_L \leftarrow (SP)$ — No flags affected

$qq_H \leftarrow (SP + 1)$

$SP \leftarrow SP + 2$

7	6	5	4	3	2	1	0
1	1	rs		0	0	0	1

POP BC
POP DE
POP HL
POP AF

Timing: M cycles—3
T states—10 (4, 3, 3)

Addressing Mode: Source—Register Indirect (Stack)
Destination—Register

7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
1	1	1	0	0	0	0	1

POP IX (for $N_X = 0$)
POP IY (for $N_X = 1$)

Timing: M cycles—4
T states—14 (4, 4, 3, 3)

Addressing Mode: Source—Register Indirect (Stack)
Destination—Register

12.6 8-Bit Arithmetic

REGISTER ADDRESSING ARITHMETIC

Op	C Before DAA	Hex Value In Upper Digit (Bits 7-4)	H Before DAA	Hex Value In Lower Digit (Bits 3-0)	Number Added To Byte	C After DAA
	0	0-9	0	0-9	00	0
	0	0-8	0	A-F	06	0
	0	0-9	1	0-3	06	0
ADD	0	A-F	0	0-9	60	1
ADC	0	9-F	0	A-F	66	1
INC	0	A-F	1	0-3	66	1
	1	0-2	0	0-9	60	1
	1	0-2	0	A-F	66	1
	1	0-3	1	0-3	66	1
SUB	0	0-9	0	0-9	00	0
SBC	0	0-8	1	6-F	FA	0
DEC	1	7-F	0	0-9	A0	1
NEG	1	6-F	1	6-F	9A	1

ADD A, r

Add contents of register r to the Accumulator.

A ← A+r

S: Set if negative result
Z: Set if zero result
H: Set if carry from bit 3
P/V: Set according to overflow condition
N: Reset
C: Set if carry from bit 7

7	6	5	4	3	2	1	0
1	0	0	0	0	r		

Timing: M cycles—1
T states—4
Addressing Mode: Source—Register
Destination—Implied

ADC A, r

Add contents of register r, plus the carry flag, to the Accumulator.

A ← A + r + CY

S: Set if negative result
Z: Set if zero result
H: Set if carry from bit 3
P/V: Set if result exceeds 2's complement range
N: Reset
C: Set if carry from bit 7

7	6	5	4	3	2	1	0
1	0	0	0	1	r		

Timing: M cycles—1
T states—4
Addressing Mode: Source—Register
Destination—Implied

SUB r

Subtract the contents of register r from the Accumulator.

A ← A − r

S: Set if result is negative
Z: Set if result is zero
H: Set if borrow from bit 4
P/V: Set if result exceeds 8-bit 2's complement range
N: Set
C: Set according to borrow

7	6	5	4	3	2	1	0
1	0	0	1	0	r		

Timing: M cycles—1
T states—4
Addressing Mode: Source—Register
Destination—Implied

SBC A, r

Subtract contents of register r and the carry bit C from the Accumulator.

A ← A − r − CY

S: Set if result is negative
Z: Set if result is zero
H: Set if borrow from bit 4
P/V: Set if result exceeds 8-bit 2's complement range
N: Set
C: Set according to borrow

7	6	5	4	3	2	1	0
1	0	0	1	1	r		

Timing: M cycles—1
T states—4
Addressing Mode: Source—Register
Destination—Implied

AND r

Logically AND the contents of the r register and the Accumulator.

A ← A ∧ r

S: Set if result is negative
Z: Set if result is zero
H: Set
P/V: Set if result parity is even
N: Reset
C: Reset

12.6 8-Bit Arithmetic (Continued)

7	6	5	4	3	2	1	0
1	0	1	0	0	r		

Timing: M cycles—1
T states—4

Addressing Mode: Source—Register
Destination—Implied

OR r

Logically OR the contents of the r register and the Accumulator.

A ← A ∨ r

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Reset

7	6	5	4	3	2	1	0
1	0	1	1	0	r		

Timing: M cycles—1
T states—4

Addressing Mode: Source—Register
Destination—Implied

XOR r

Logically exclusively OR the contents of the r register with the Accumulator.

A ← A ⊕ r

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Reset

7	6	5	4	3	2	1	0
1	0	1	0	1	r		

Timing: M cycles—1
T states—4

Addressing Mode: Source—Register
Destination—Implied

INC r

Increment register r.

r ← r + 1

S: Set if result is negative
Z: Set if result is zero
H: Set if carry from bit 3
P/V: Set only if r was X'7F before operation
N: Reset
C: N/A

7	6	5	4	3	2	1	0
0	0	r			1	0	0

Timing: M cycles—1
T states—4

Addressing Mode: Source—Register
Destination—Register

CP r

Compare the contents of register r with the Accumulator and set the flags accordingly.

A − r

S: Set if result is negative
Z: Set if result is zero
H: Set if borrow from bit 4
P/V: Set if result exceeds 8-bit 2's complement range
N: Set
C: Set according to borrow

7	6	5	4	3	2	1	0
1	0	1	1	1	r		

Timing: M cycles—1
T states—4

Addressing Mode: Source—Register
Destination—Implied

DEC r

Decrement the contents of register r.

r ← r − 1

S: Set if result is negative
Z: Set if result is zero
H: Set according to a borrow from bit 4
P/V: Set only if r was X'80 prior to operation
N: Set
C: N/A

7	6	5	4	3	2	1	0
0	0	r			1	0	1

Timing: M cycles—1
T states—4

Addressing Mode: Source—Register
Destination—Register

CPL

Complement the Accumulator (1's complement).

A ← $\bar{A}$

S: N/A
Z: N/A
H: Set
P/V: N/A
N: Set
C: N/A

12.6 8-Bit Arithmetic (Continued)

7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	1

Timing: M cycles—1
T states—4
Addressing Mode: Implied

NEG

Negate the Accumulator (2's complement).

A ← 0 − A
S: Set if result is negative
Z: Set if result is zero
H: Set according to borrow from bit 4
P/V: Set only if Accumulator was X'80 prior to operation
N: Set
C: Set only if Accumulator was not X'00 prior to operation

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	0	0	0	1	0	0

Timing: M cycles—2
T states—8 (4, 4)
Addressing Mode: Implied

CCF

Complement the carry flag.

CY ← $\overline{CY}$
S: N/A
Z: N/A
H: Previous carry
P/V: N/A
N: Reset
C: Complement of previous carry

7	6	5	4	3	2	1	0
0	0	1	1	1	1	1	1

Timing: M cycles—1
T states—4
Addressing Mode: Implied

SCF

Set the carry flag.

CY ← 1
S: N/A
Z: N/A
H: Reset
P/V: N/A
N: Reset
C: Set

7	6	5	4	3	2	1	0
0	0	1	1	0	1	1	1

Timing: M cycles—1
T states—4
Addressing Mode: Implied

DAA

Adjust the Accumulator for BCD addition and subtraction operations. To be executed after BCD data has been operated upon the standard binary ADD, ADC, INC, SUB, SBC, DEC or NEG instructions (see "Register Addressing Arithmetic" table).

——
S: Set according to bit 7 of result
Z: Set if result is zero
H: Set according to instructions
P/V: Set according to parity of result
N: N/A
C: Set according to instructions

7	6	5	4	3	2	1	0
0	0	1	0	0	1	1	1

Timing: M cycles—1
T states—4
Addressing Mode: Implied

IMMEDIATELY ADDRESSED ARITHMETIC

ADD A, n

Add the immediate data n to the Accumulator.

A ← A + n
S: Set if result is negative
Z: Set if result is zero
H: Set if carry from bit 3
P/V: Set if result exceeds 8-bit 2's complement range
N: Reset
C: Set if carry from bit 7

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

Timing: M cycles—2
T states—7 (4, 3)
Addressing Mode: Source—Immediate
Destination—Implied

ADC A, n

Add, with carry, the immediate data n and the Accumulator.

A ← A + n + CY
S: Set if result is negative
Z: Set if result is zero
H: Set if carry from bit 3
P/V: Set if result exceeds 8-bit 2's complement range
N: Reset
C: Set according to carry from bit 7

12.6 8-Bit Arithmetic (Continued)

7	6	5	4	3	2	1	0
1	1	0	0	1	1	1	0

n

Timing: M cycles—2
T states—7 (4, 3)
Addressing Mode: Source—Immediate
Destination—Implied

SUB n

Subtract the immediate data n from the Accumulator.

A ← A − n

S: Set if result is negative
Z: Set if result is zero
H: Set if borrow from bit 4
P/V: Set if result exceeds 8-bit 2's complement range
N: Set
C: Set according to borrow condition

7	6	5	4	3	2	1	0
1	1	0	1	0	1	1	0

n

Timing: M cycles—2
T states—7 (4, 3)
Addressing Mode: Source—Immediate
Destination—Implied

SBC A, n

Subtract, with carry, the immediate data n from the Accumulator.

A ← A − n − CY

S: Set if result is negative
Z: Set if result is zero
H: Set if borrow from bit 4
P/V: Set if result exceeds 8-bit 2's complement range
N: Set
C: Set according to borrow condition

7	6	5	4	3	2	1	0
1	1	0	1	1	1	1	0

n

Timing: M cycles—2
T states—7 (4, 3)
Addressing Mode: Source—Immediate
Destination—Implied

AND n

The immediate data n is logically AND'ed to the Accumulator.

A ← A ∧ n

S: Set if result is negative
Z: Set if result is zero
H: Set
P/V: Set if result parity is even
N: Reset
C: Reset

7	6	5	4	3	2	1	0
1	1	1	0	0	1	1	0

n

Timing: M cycles—2
T states—7 (4, 3)
Addressing Mode: Source—Immediate
Destination—Implied

OR n

The immediate data n is logically OR'ed to the contents of the Accumulator.

A ← A ∨ s

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Reset

7	6	5	4	3	2	1	0
1	1	1	1	0	1	1	0

n

Timing: M cycles—2
T states—7 (4, 3)
Addressing Mode: Source—Immediate
Destination—Implied

XOR n

The immediate data n is exclusively OR'ed with the Accumulator.

A ← A ⊕ n

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Reset

12.6 8-Bit Arithmetic (Continued)

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

Timing: M cycles—2
T states—7 (4, 3)

Addressing Mode: Source—Immediate
Destination—Implied

CP n

Compare the immediate data n with the contents of the Accumulator via subtraction and return the appropriate flags. The contents of the Accumulator are not affected.

A − n

S: Set if result is negative
Z: Set if result is zero
H: Set if borrow from bit 4
P/V: Set if result exceeds 8-bit 2's complement range
N: Set
C: Set according to borrow condition

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

Timing: M cycles—2
T states—7 (4, 3)

Addressing Mode: Immediate

MEMORY ADDRESSED ARITHMETIC

ADD A, m1

Add the contents of the memory location m_1 to the Accumulator.

$A \leftarrow A + m_1$

S: Set if result is negative
Z: Set if result is zero
H: Set if carry from bit 3
P/V: Set if result exceeds 8-bit 2's complement range
N: Reset
C: Set according to carry from bit 7

7	6	5	4	3	2	1	0	
1	0	0	0	0	1	1	0	ADD A, (HL)

Timing: M cycles—2
T states—7 (4, 3)

Addressing Mode: Source—Register Indirect
Destination—Implied

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	ADD A, (IX + d) (for N_X=0) ADD A, (IY + d) (for N_X=1)
1	0	0	0	0	1	1	0	
d								

Timing: M cycles—5
T states—19 (4, 4, 3, 5, 3)

Addressing Mode: Source—Indexed
Destination—Implied

ADC A, m_1

Add the contents of the memory location m_1 plus the carry to the Accumulator.

$A \leftarrow A + m_1 + CY$

S: Set if result is negative
Z: Set if result is zero
H: Set if carry from bit 3
P/V: Set if result exceeds 8-bit 2's complement range
N: Reset
C: Set according to carry from bit 7

7	6	5	4	3	2	1	0	
1	0	0	0	1	1	1	0	ADC A, (HL)

Timing: M cycles—2
T states—7 (4, 3)

Addressing Mode: Source—Register Indirect
Destination—Implied

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	ADC A, (IX + d) (for N_X=0) ADC A, (IY + d) (for N_X=1)
1	0	0	0	1	1	1	0	
d								

Timing: M cycles—5
T states—19 (4, 4, 3, 5, 3)

Addressing Mode: Source—Indexed
Destination—Implied

SUB m_1

Subtract the contents of memory location m_1 from the Accumulator.

$A \leftarrow A - m_1$

S: Set if result is negative
Z: Set if result is zero
H: Set if borrow from bit 4
P/V: Set if result exceeds 8-bit 2's complement range
N: Set
C: Set according to borrow condition

12.6 8-Bit Arithmetic (Continued)

7	6	5	4	3	2	1	0
1	0	0	1	0	1	1	0

SUB (HL)

Timing: M cycles—2
T states—7 (4, 3)

Addressing Mode: Source—Register Indirect
Destination—Implied

7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
1	0	0	1	0	1	1	0
d							

SUB (IX + d) (for $N_X = 0$)
SUB (IY + d) (for $N_X = 1$)

Timing: M cycles—5
T states—19 (4, 4, 3, 5, 3)

Addressing Mode: Source—Indexed
Destination—Implied

SBC A, m_1

Subtract, with carry, the contents of memory location m_1 from the Accumulator.

$A \leftarrow A - m_1 - CY$

S: Set if result is negative
Z: Set if result is zero
H: Set if carry from bit 3
P/V: Set if result exceeds 8-bit 2's complement range
N: Set
C: Set according to borrow condition

7	6	5	4	3	2	1	0
1	0	0	1	1	1	1	0

SBC A, (HL)

Timing: M cycles—2
T states—7 (4, 3)

Addressing Mode: Source—Register Indirect
Destination—Implied

7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
1	0	0	1	1	1	1	0
d							

SBC A, (IX + d) (for $N_X = 0$)
SBC A, (IY + d) (for $N_X = 1$)

Timing: M cycles—5
T states—19 (4, 4, 3, 5, 3)

Addressing Mode: Source—Indexed
Destination—Implied

AND m_1

The data in memory location m_1 is logically AND'ed to the Accumulator.

$A \leftarrow A \wedge m_1$

S: Set if result is negative
Z: Set if result is zero
H: Set
P/V: Set if result parity is even
N: Reset
C: Reset

7	6	5	4	3	2	1	0
1	0	1	0	0	1	1	0

AND (HL)

Timing: M cycles—2
T states—7 (4, 3)

Addressing Mode: Source—Register Indirect
Destination—Implied

7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
1	0	1	0	0	1	1	0
d							

AND (IX + d) (for $N_X = 0$)
AND (IY + d) (for $N_X = 1$)

Timing: M cycles—5
T states—19 (4, 4, 3, 5, 3)

Addressing Mode: Source—Indexed
Destination—Implied

OR m_1

The data in memory location m_1 is logically OR'ed with the Accumulator.

$A \leftarrow A \vee m_1$

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Reset

7	6	5	4	3	2	1	0
1	0	1	1	0	1	1	0

OR (HL)

Timing: M cycles—2
T states—7 (4, 3)

Addressing Mode: Source—Register Indexed
Destination—Implied

7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
1	0	1	1	0	1	1	0
d							

OR (IX + d) (for $N_X = 0$)
OR (IY + d) (for $N_X = 1$)

Timing: M cycles—5
T states—19 (4, 4, 3, 5, 3)

Addressing Mode: Source—Indexed
Destination—Implied

12.6 8-Bit Arithmetic (Continued)

XOR m_1

The data in memory location m_1 is exclusively OR'ed with the data in the Accumulator.

$A \leftarrow A \oplus m_1$

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Reset

7	6	5	4	3	2	1	0	
1	0	1	0	1	1	1	0	XOR (HL)

Timing: M cycles—2
T states—7 (4, 3)
Addressing Mode: Source—Register Indexed
Destination—Implied

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	XOR (IX + d) (for $N_X = 0$) / XOR (IY + d) (for $N_X = 1$)
1	0	1	0	1	1	1	0	
d								

Timing: M cycles—5
T states—19 (4, 4, 3, 5, 3)
Addressing Mode: Source—Indexed
Destination—Implied

CP m_1

Compare the data in memory location m_1 with the data in the Accumulator via subtraction.

$A - m_1$

S: Set if result is negative
Z: Set if result is zero
H: Set if borrow from bit 4
P/V: Set if result exceeds 8-bit 2's complement range
N: Set
C: Set according to borrow condition

7	6	5	4	3	2	1	0	
1	0	1	1	1	1	1	0	CP (HL)

Timing: M cycles—2
T states—7 (4, 3)
Addressing Mode: Source—Register Indirect
Destination—Implied

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	CP (IX + d) (for $N_X = 0$) / CP (IY + d) (for $N_X = 1$)
1	0	1	1	1	1	1	0	
d								

Timing: M cycles—5
T states—19 (4, 4, 3, 5, 3)
Addressing Mode: Source—Indexed
Destination—Implied

INC m_1

Increment data in memory location m_1.

$m_1 \leftarrow m_1 + 1$

S: Set if result is negative
Z: Set if result is zero
H: Set according to carry from bit 3
P/V: Set if data was X'7F before operation
N: Reset
C: N/A

7	6	5	4	3	2	1	0	
0	0	1	1	0	1	0	0	INC (HL)

Timing: M cycles—3
T states—11 (4, 4, 3)
Addressing Mode: Source—Register Indexed
Destination—Register Indexed

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	INC (IX + d) (for $N_X = 0$) / INC (IY + d) (for $N_X = 1$)
0	0	1	1	0	1	0	0	
d								

Timing: M cycles—6
T states—23 (4, 4, 3, 5, 4, 3)
Addressing Mode: Source—Indexed
Destination—Indexed

DEC m_1

Decrement data in memory location m_1.

$m_1 \leftarrow m_1 - 1$

S: Set if result is negative
Z: Set if result is zero
H: Set according to borrow from bit 4
P/V: Set only if m_1 was X'80 before operation
N: Set
C: N/A

12.6 8-Bit Arithmetic (Continued)

7	6	5	4	3	2	1	0	
0	0	1	1	0	1	0	1	DEC (HL)

Timing: M cycles — 3
T states — 11 (4, 4, 3)
Addressing Mode: Source — Register Indexed
Destination — Register Indexed

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	DEC (IX + d) (for N_X = 0) DEC (IY + d) (for N_X = 1)
0	0	1	1	0	1	0	1	
d								

Timing: M cycles — 6
T states — 23 (4, 4, 3, 5, 4, 3)
Addressing Mode: Source — Indexed
Destination — Indexed

12.7 16-Bit Arithmetic

ADD ss, pp

Add the contents of the 16-bit register rp or pp to the contents of the 16-bit register ss.

ss ← ss + rp
or
ss ← ss + pp

S: N/A
Z: N/A
H: Set if carry from bit 11
P/V: N/A
N: Reset
C: Set if carry from bit 15

7	6	5	4	3	2	1	0	
0	0	rp		1	0	0	1	ADD HL, rp

Timing: M cycles — 3
T states — 11 (4, 4, 3)
Addressing Mode: Source — Register
Destination — Register

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	ADD IX, pp (for N_X = 0) ADD IY, pp (for N_X = 1)
0	0	pp		1	0	0	1	

Timing: M cycles — 4
T states — 15 (4, 4, 4, 3)
Addressing Mode: Source — Register
Destination — Register

ADC HL, pp

The contents of the 16-bit register pp are added, with the carry bit, to the HL register.

HL ← HL + pp + CY

S: Set if result is negative
Z: Set if result is zero
H: Set according to carry out of bit 11
P/V: Set if result exceeds 16-bit 2's complement range
N: Reset
C: Set if carry out of bit 15

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	pp		1	0	1	0

Timing: M cycles — 4
T states — 15 (4, 4, 4, 3)
Addressing Mode: Source — Register
Destination — Register

SBC HL, pp

Subtract, with carry, the contents of the 16-bit pp register from the 16-bit HL register.

HL ← HL − pp − CY

S: Set if result is negative
Z: Set if result is zero
H: Set according to borrow from bit 12
P/V: Set if result exceeds 16-bit 2's complement range
N: Set
C: Set according to borrow condition

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	pp		0	0	1	0

Timing: M cycles — 4
T states — 15 (4, 4, 4, 3)
Addressing Mode: Source — Register
Destination — Register

INC rr

Increment the contents of the 16-bit register rr.

rr ← rr + 1 No flags affected

7	6	5	4	3	2	1	0	
0	0	rp		0	0	1	1	INC BC INC DE INC HL INC SP

Timing: M cycles — 1
T states — 6
Addressing Mode: Register

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	INC IX (for N_X=0) INC IY (for N_X=1)
0	0	1	0	0	0	1	1	

Timing: M cycles — 2
T states — 10 (4, 6)
Addressing Mode: Register

12.7 16-Bit Arithmetic (Continued)

DEC rr

Decrement the contents of the 16-bit register rr.

rr ← rr − 1 — No flags affected

7	6	5	4	3	2	1	0
0	0	rp		1	0	1	1

DEC BC
DEC DE
DEC HL
DEC SP

Timing: M cycles — 1
T states — 6

Addressing Mode: Register

7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
0	0	1	0	1	0	1	1

DEC IX (for $N_X=0$)
DEC IY (for $N_X=1$)

Timing: M cycles — 2
T states — 10 (4, 6)

Addressing Mode: Register

12.8 Bit Set, Reset, and Test

REGISTER

SET b, r

Bit b in register r is set.

R_b ← 1 — No flags affected

7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	1
1	1	b			r		

Timing: M cycles — 2
T states — 8 (4, 4)

Addressing Mode: Bit/Register

RES b, r

Bit b in register r is reset.

r_b ← 0 — No flags affected

7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	1
1	0	b			r		

Timing: M cycles — 2
T states — 8 (4, 4)

Addressing Mode: Bit/Register

BIT b, r

Bit b in register r is tested with the result put in the Z flag.

Z ← $\overline{r_b}$

S: Undefined
Z: Inverse of tested bit
H: Set
P/V: Undefined
N: Reset
C: N/A

7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	1
0	1	b			r		

Timing: M cycles — 2
T states — 8 (4, 4)

Addressing Mode: Bit/Register

MEMORY

SET b, m_1

Bit b in memory location m_1 is set.

m_{1b} ← 1 — No flags affected

7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	1
1	1	b			1	1	0

SET b, (HL)

Timing: M cycles — 4
T states — 15 (4, 4, 4, 3)

Addressing Mode: Bit/Register Indirect

7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
1	1	0	0	1	0	1	1
d							
1	1	b			1	1	0

SET b, (IX+d) (for $N_X=0$)
SET b, (IY+d) (for $N_X=1$)

Timing: M cycles — 6
T states — 23 (4, 4, 3, 5, 4, 3)

Addressing Mode: Bit/Indexed

RES b, m_1

Bit b in memory location m_1 is reset.

m_{1b} ← 0 — No flags affected

7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	1
1	0	b			1	1	0

RES b, (HL)

Timing: M cycles — 4
T states — 15 (4, 4, 4, 3)

Addressing Mode: Bit/Register Indirect

7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
1	1	0	0	1	0	1	1
d							
1	0	b			1	1	0

RES b, (IX+d) (for $N_X=0$)
RES b, (IY+d) (for $N_X=1$)

Timing: M cycles — 6
T states — 23 (4, 4, 3, 5, 4, 3)

Addressing Mode: Bit/Indexed

12.8 Bit Set, Reset, and Test (Continued)

BIT B, m_1

Bit b in memory location m_1 is tested via the Z flag.

$Z \leftarrow \overline{m_{1b}}$

S: Undefined
Z: Inverse of tested bit
H: Set
P/V: Undefined
N: Reset
C: N/A

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	BIT b, (HL)
0	1	b			1	1	0	

Timing: M cycles — 3
T states — 12 (4, 4, 4)

Addressing Mode: Bit/Register Indirect

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	BIT b, (IX+d) (for $N_X=0$) BIT b, (IY+d) (for $N_X=1$)
1	1	0	0	1	0	1	1	
d								
0	1	b			1	1	0	

Timing: M cycles — 5
T states — 20 (4, 4, 3, 5, 4)

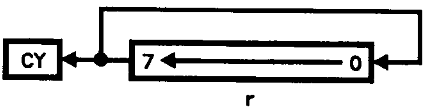
Addressing Mode: Bit/Indexed

12.9 Rotate and Shift

REGISTER

RLC r

Rotate register r left circular.



TL/C/5171-57

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Set according to bit 7 of r

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	RLC r
0	0	0	0	0	r			(Note alternate for A register below)

Timing: M cycles — 2
T states — 8 (4, 4)

Addressing Mode: Register

7	6	5	4	3	2	1	0	
0	0	0	0	0	1	1	1	RLCA

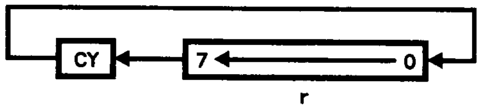
Timing: M cycles — 1
T states — 4

Addressing Mode: Implied

(Note RLCA does not affect S, Z, or P/V flags.)

RL r

Rotate register r left through carry.



TL/C/5171-58

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Set according to bit 7 of r

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	RL r
0	0	0	1	0	r			(Note alternate for A register below)

Timing: M cycles — 2
T states — 8 (4, 4)

Addressing Mode: Register

7	6	5	4	3	2	1	0	
0	0	0	1	0	1	1	1	RLA

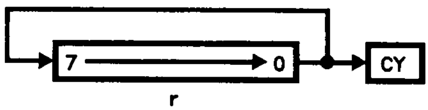
Timing: M cycles — 1
T states — 4

Addressing Mode: Implied

(Note RLA does not affect S, Z, or P/V flags.)

RRC r

Rotate register r right circular.



TL/C/5171-59

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Set according to bit 0 of r

12.9 Rotate and Shift (Continued)

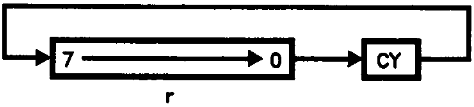
7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	RRC r
0	0	0	0	1	r			(Note alternate for A register below)

Timing: M cycles — 2
T states — 8 (4, 4)
Addressing Mode: Register

7	6	5	4	3	2	1	0	
0	0	0	0	1	1	1	1	RRCA

Timing: M cycles — 1
T states — 4
Addressing Mode: Implied
(Note RRCA does not affect S, Z, or P/V flags.)

RR r
Rotate register r right through carry.



TL/C/5171–60

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Set according to bit 0 of r

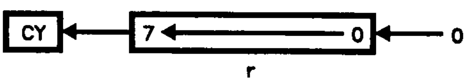
7	6	5	4	3	2	1	0	
1	1	0	0	1	0	0	1	RR r
0	0	0	1	1	r			(Note alternate for A register below)

Timing: M cycles — 2
T states — 8 (4, 4)
Addressing Mode: Register

7	6	5	4	3	2	1	0	
0	0	0	1	1	1	1	1	RRA

Timing: M cycles — 1
T states — 4
Addressing Mode: Implied
(Note RRA does not affect S, Z, or P/V flags.)

SLA r
Shift register r left arithmetic.



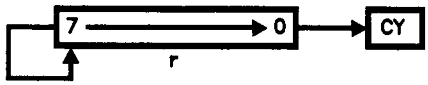
TL/C/5171–61

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Set according to bit 7 of r

7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	1
0	0	1	0	0	r		

Timing: M cycles — 2
T states — 8 (4, 4)
Addressing Mode: Register

SRA r
Shift register r right arithmetic.



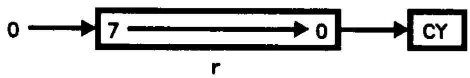
TL/C/5171–62

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Set according to bit 0 of r

7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	1
0	0	1	0	1	r		

Timing: M cycles — 2
T states — 8 (4, 4)
Addressing Mode: Register

SRL r
Shift register r right logical.



TL/C/5171–63

S: Reset
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Set according to bit 0 of r

7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	1
0	0	1	1	1	r		

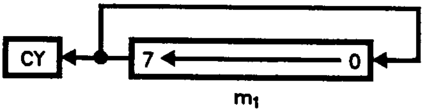
Timing: M cycles — 2
T states — 8 (4, 4)
Addressing Mode: Register

12.9 Rotate and Shift (Continued)

MEMORY

RLC m_1

Rotate date in memory location m_1 left circular.



TL/C/5171-64

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Set according to bit 7 of m_1

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	RLC (HL)
0	0	0	0	0	1	1	0	

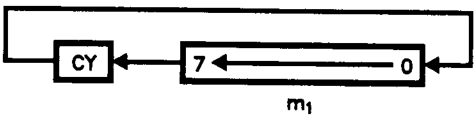
Timing: M cycles — 4
T states — 15 (4, 4, 4, 3)
Addressing Mode: Register indirect

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	RLC (IX+d) (for N_X=0) RLC (IY+d) (for N_X=1)
1	1	0	0	1	0	1	1	
d								
0	0	0	0	0	1	1	0	

Timing: M cycles — 6
T states — 23 (4, 4, 3, 5, 4, 3)
Addressing Mode: Indexed

RL m_1

Rotate the data in memory location m_1 left though carry.



TL/C/5171-65

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Set according to bit 7 of m_1

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	RL (HL)
0	0	0	1	0	1	1	0	

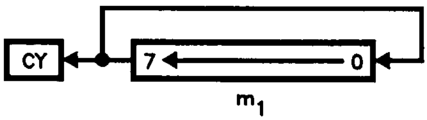
Timing: M cycles — 4
T states — 15 (4, 4, 4, 3)
Addressing Mode: Register Indirect

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	RL (IX+d) (for N_X=0) RL (IY+d) (for NX=1)
1	1	0	0	1	0	1	1	
d								
0	0	0	1	0	1	1	0	

Timing: M cycles — 6
T states — 23 (4, 4, 3, 5, 4, 3)
Addressing Mode: Indexed

RRC m_1

Rotate the data in memory location m_1 right circular.



TL/C/5171-66

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Set according to bit 0 of m_1

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	RRC (HL)
0	0	0	0	1	1	1	0	

Timing: M cycles — 4
T states — 15 (4, 4, 4, 3)
Addressing Mode: Register Indirect

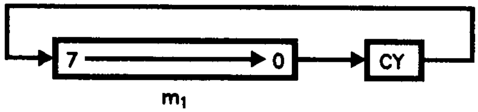
7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	RRC (IX + d) (for N_X = 0) RRC (IY + d) (for N_X = 1)
1	1	0	0	1	0	1	1	
d								
0	0	0	0	1	1	1	0	

Timing: M cycles — 6
T states — 23 (4, 4, 3, 5, 4, 3)
Addressing Mode: Indexed

12.9 Rotate and Shift (Continued)

RR m_1

Rotate the data in memory location m_1 right through the carry.



TL/C/5171-67

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Set according to bit 0 of m_1

7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	1
0	0	0	1	1	1	1	0

RR (HL)

Timing: M cycles — 4
T states — 15 (4, 4, 4, 3)
Addressing Mode: Register Indirect

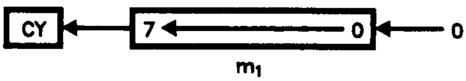
7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
1	1	0	0	1	0	1	1
d							
0	0	0	1	1	1	1	0

RR (IX + d) (for $N_X = 0$)
RR (IY + d) (for $N_X = 1$)

Timing: M cycles — 6
T states — 23 (4, 4, 3, 5, 4, 3)
Addressing Mode: Indexed

SLA m_1

Shift the data in memory location m_1 left arithmetic.



TL/C/5171-68

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Set according to bit 7 of m_1

7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	1
0	0	1	0	0	1	1	0

SLA (HL)

Timing: M cycles — 4
T states — 15 (4, 4, 4, 3)
Addressing Mode: Register Indirect

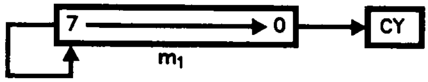
7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
1	1	0	0	1	0	1	1
d							
0	0	1	0	0	1	1	0

SLA (IX + d) (for $N_X = 0$)
SLA (IY + d) (for $N_X = 1$)

Timing: M cycles — 6
T states — 23 (4, 4, 3, 5, 4, 3)
Addressing Mode: Indexed

SRA m_1

Shift the data in memory location m_1 right arithmetic.



TL/C/5171-69

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Set according to bit 0 of m_1

7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	1
0	0	1	0	1	1	1	0

SRA (HL)

Timing: M cycles — 4
T states — 15 (4, 4, 4, 3)
Addressing Mode: Register Indirect

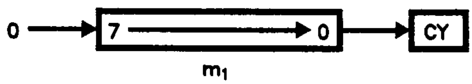
7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
1	1	0	0	1	0	1	1
d							
0	0	1	0	1	1	1	0

SRA (IX + d) (for $N_X = 0$)
SRA (IY + d) (for $N_X = 1$)

Timing: M cycles — 6
T states — 23 (4, 4, 3, 5, 4, 3)
Addressing Mode: Indexed

SRL m_1

Shift right logical the data in memory location m_1.



TL/C/5171-70

S: Reset
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: Set according to bit 0 of m_1

12.9 Rotate and Shift (Continued)

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	SRL (HL)
0	0	1	1	1	1	1	0	

Timing: M cycles — 4
T states — 15 (4, 4, 4, 3)
Addressing Mode: Register Indirect

7	6	5	4	3	2	1	0
1	1	N_X	1	1	1	0	1
1	1	0	0	1	0	1	1
d							
0	0	1	1	1	1	1	0

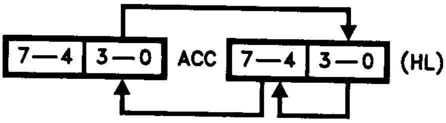
SRL (IX + d) (for $N_X = 0$)
SRL (IY + d) (for $N_X = 1$)

Timing: M cycles — 6
T states — 23 (4, 4, 3, 5, 4, 3)
Addressing Mode: Indexed

REGISTER/MEMORY

RLD

Rotate digit left and right between the Accumulator and memory (HL).



TL/C/5171-71

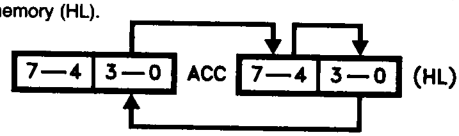
S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	1	0	1	1	1	1

Timing: M cycles — 5
T states — 18 (4, 4, 3, 4, 3)
Addressing Mode: Implied/Register Indirect

RRD

Rotate digit right and left between the Accumulator and memory (HL).



TL/C/5171-72

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	1	0	0	1	1	1

Timing: M cycles — 5
T states — 18 (4, 4, 3, 4, 3)
Addressing Mode: Implied/Register Indirect

12.10 Exchanges

REGISTER/REGISTER

EX DE, HL

Exchange the contents of the 16-bit register pairs DE and HL.

DE ⟷ HL — No flags affected

7	6	5	4	3	2	1	0
1	1	1	0	1	0	1	1

Timing: M cycles — 1
T states — 4
Addressing Mode: Register

EX AF, A'F'

The contents of the Accumulator and flag register are exchanged with their corresponding alternate registers, that is A and F are exchanged with A' and F'.

A ⟷ A' — No flags affected
F ⟷ F'

7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0

Timing: M cycles — 1
T states — 4
Addressing Mode: Register

12.10 Exchanges (Continued)

EXX

Exchange the contents of the BC, DE, and HL registers with their corresponding alternate register.

BC ⟷ B'C' — No flags affected
DE ⟷ D'E'
HL ⟷ H'L'

7	6	5	4	3	2	1	0
1	1	0	1	1	0	0	1

Timing: M cycles — 1
T states — 4

Addressing Mode: Implied

REGISTER/MEMORY

EX (SP), ss

Exchange the two bytes at the top of the external memory stack with the 16-bit register ss.

(SP) ⟷ SS_L — No flags affected
(SP + 1) ⟷ SS_H

7	6	5	4	3	2	1	0	
1	1	1	0	0	0	1	1	EX (SP), HL

Timing: M cycles — 5
T states — 19 (4, 3, 4, 3, 5)

Addressing Mode: Register/Register Indirect

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	EX (SP), IX (for N_X = 0)
1	1	1	0	0	0	1	1	EX (SP),IY (for N_X = 1)

Timing: M cycles — 6
T states — 23 (4, 4, 3, 4, 3, 5)

Addressing Mode: Register/Register Indirect

12.11 Memory Block Moves and Searches

SINGLE OPERATIONS

LDI

Move data from memory location (HL) to memory location (DE), increment memory pointers, and decrement byte counter BC.

(DE) ← (HL)
DE ← DE + 1
HL ← HL + 1
BC ← BC − 1

S: N/A
Z: N/A
H: Reset
P/V: Set if BC −1 ≠0, otherwise reset
N: Reset
C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	0	0	0	0	0

Timing: M cycles — 4
T states — 16 (4, 4, 3, 5)

Addressing Mode: Register Indirect

LDD

Move data from memory location (HL) to memory location (DE), and decrement memory pointer and byte counter BC.

(DE) ← (HL)
DE ← DE − 1
HL ← HL − 1
BC ← BC − 1

S: N/A
Z: N/A
H: Reset
P/V: Set if BC −1 ≠0, otherwise reset
N: Reset
C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	0	1	0	0	0

Timing: M cycles — 4
T states — 16 (4, 4, 3, 5)

Addressing Mode: Register Indirect

CPI

Compare data in memory location (HL) to the Accumulator, increment the memory pointer, and decrement the byte counter. The Z flag is set if the comparison is equal.

A − (HL)
HL ← HL + 1
BC ← BC − 1
Z ← 1
if A = (HL)

S: Set if result of comparison subtract is negative
Z: Set if result of comparison is zero
H: Set according to borrow from bit 4
P/V: Set if BC − 1≠ 0, otherwise reset
N: Set
C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	0	0	0	0	1

Timing: M cycles — 4
T states — 16 (4, 4, 3, 5)

Addressing Mode: Register Indirect

CPD

Compare data in memory location (HL) to the Accumulator, and decrement the memory pointer and byte counter. The Z flag is set if the comparison is equal.

A − (HL)
HL ← HL − 1
BC ← BC − 1
Z ← 1
if A = (HL)

S: Set if result is negative
Z: Set if result of comparison is zero
H: Set according to borrow from bit 4
P/V: Set if BC − 1≠ 0, otherwise reset
N: Set
C: N/A

12.11 Memory Block Moves and Searches (Continued)

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	0	1	0	0	1

Timing: M cycles — 4
T states — 16 (4, 4, 3, 5)

Addressing Mode: Register Indirect

REPEAT OPERATIONS

LDIR

Move data from memory location (HL) to memory location (DE), increment memory pointers, decrement byte counter BC, and repeat until BC = 0.

(DE) ← (HL)	S: N/A
DE ← DE + 1	Z: N/A
HL ← HL + 1	H: Reset
BC ← BC − 1	P/V: Reset
Repeat until	N: Reset
BC = 0	C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	1	0	0	0	0

Timing: For BC≠0 M cycles — 5
T states — 21 (4, 4, 3, 5, 5)
For BC=0 M cycles — 4
T states — 16 (4, 4, 3, 5)

Addressing Mode: Register Indirect

(Note that each repeat is accomplished by a decrement of the BC, so that refresh, etc. continues for each cycle.)

LDDR

Move data from memory location (HL) to memory location (DE), decrement memory pointers and byte counter BC, and repeat until BC = 0.

(DE) ← (HL)	S: N/A
DE ← DE − 1	Z: N/A
HL ← HL − 1	H: Reset
BC ← BC − 1	P/V: Reset
Repeat until	N: Reset
BC = 0	C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	1	1	0	0	0

Timing: For BC≠0 M cycles — 5
T states — 21 (4, 4, 3, 5, 5)
For BC=0 M cycles — 4
T states — 16 (4, 4, 3, 5)

Addressing Mode: Register Indirect

(Note that each repeat is accomplished by a decrement of the BC, so that refresh, etc. continues for each cycle.)

CPIR

Compare data in memory location (HL) to the Accumulator, increment the memory, decrement the byte counter BC, and repeat until BC = 0 or (HL) equals A.

A − (HL)	S: Set if sign of subtraction performed for comparison is negative
HL ← HL + 1	Z: Set if A = (HL), otherwise reset
BC ← BC − 1	H: Set according to borrow from bit 4
Repeat until BC = 0 or A = (HL)	P/V: Set if BC − 1 ≠ 0, otherwise reset
	N: Set
	C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	1	0	0	0	1

Timing: For BC ≠ 0 M cycles — 5
T states — 21 (4, 4, 3, 5, 5)
For BC = 0 M cycles — 4
T states — 16 (4, 4, 3, 5)

Addressing Mode: Register Indirect

(Note that each repeat is accomplished by a decrement of the PC, so that refresh, etc. continues for each cycle.)

CPDR

Compare data in memory location (HL) to the contents of the Accumulator, decrement the memory pointer and byte counter BC, and repeat until BC = 0, or until (HL) equals the Accumulator.

A − (HL)	S: Set if sign of subtraction performed for comparison is negative
HL ← HL − 1	Z: Set according to equality of A and (HL), set if true
BC ← BC − 1	H: Set according to borrow from bit 4
Repeat until BC = 0 or A = (HL)	P/V: Set if BC − 1 ≠ 0, otherwise reset
	N: Set
	C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	1	1	0	0	1

Timing: For BC ≠ 0 M cycles — 5
T states — 21 (4, 4, 3, 5, 5)
For BC = 0 M cycles — 4
T states — 16 (4, 4, 3, 5)

Addressing Mode: Register Indirect

(Note that each repeat is accomplished by a decrement of the BC, so that refresh, etc. continues for each cycle.)

12.12 Input/Output

IN A, (n)

Input data to the Accumulator from the I/O device at address N.

A ← (n) No flags affected

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

Timing: M cycles — 3

T states — 11 (4, 3, 4)

Addressing Mode: Source — Direct

Destination — Register

IN r, (C)

Input data to register r from the I/O device addressed by the contents of register C. If r=110 only flags are affected.

r ← (C)

S: Set if result is negative
Z: Set if result is zero
H: Reset
P/V: Set if result parity is even
N: Reset
C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	r			0	0	0

Timing: M cycles — 3

T states — 12 (4, 4, 4)

Addressing Mode: Source — Register Indirect

Destination — Register

OUT (C), r

Output register r to the I/O device addressed by the contents of register C.

(C) ← r No flags affected

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	r			0	0	1

Timing: M cycles — 3

T states — 12 (4, 4, 4)

Addressing Mode: Source — Register

Destination — Register Indirect

INI

Input data from the I/O device addressed by the contents of register C to the memory location pointed to by the contents of the HL register. The HL pointer is incremented and the byte counter B is decremented.

(HL) ← (C)
B ← B − 1
HL ← HL + 1

S: Undefined
Z: Set if B−1=0, otherwise reset
H: Undefined
P/V: Undefined
N: Set
C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	0	0	0	1	0

Timing: M cycles — 4

T states — 16 (4, 5, 3, 4)

Addressing Mode: Implied/Source — Register Indirect

Destination — Register Indirect

OUTI

Output data from memory location (HL) to the I/O device at port address (C), increment the memory pointer, and decrement the byte counter B.

(C) ← (HL)
B ← B − 1
HL ← HL + 1

S: Undefined
Z: Set if B−1=0, otherwise reset
H: Undefined
P/V: Undefined
N: Set
C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	0	0	0	1	1

Timing: M cycles — 4

T states — 16 (4, 5, 3, 4)

Addressing Mode: Implied/Source — Register Indirect

Destination — Register Indirect

IND

Input data from I/O device at port address (C) to memory location (HL), and decrement HL memory pointer and byte counter B.

(HL) ← (C)
HL ← HL − 1
B ← B − 1

S: Undefined
Z: Set if B−1=0, otherwise reset
H: Undefined
P/V: Undefined
N: Set
C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	0	1	0	1	0

Timing: M cycles — 4

T states — 16 (4, 5, 3, 4)

Addressing Mode: Implied/Source — Register Indirect

Destination — Register Indirect

12.12 Input/Output (Continued)

OUT (n), A

Output the Accumulator to the I/O device at address n.

(n) ← A   No flags affected

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

Timing: M cycles — 3
T states — 11 (4, 3, 4)

Addressing Mode: Source — Register
Destination — Direct

OUTD

Data is output from memory location (HL) to the I/O device at port address (C), and the HL memory pointer and byte counter B are decremented.

(C) ← (HL)	S: Undefined
B ← B − 1	Z: Set if B − 1 = 0, otherwise reset
HL ← HL − 1	H: Undefined
	P/V: Undefined
	N: Set
	C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	0	1	0	1	1

Timing: M cycles — 4
T states — 16 (4, 5, 3, 4)

Addressing Mode: Implied/Source — Register Indirect
Destination — Register Indirect

INIR

Data is input from the I/O device at port address (C) to memory location (HL), the HL memory pointer is incremented, and the byte counter B is decremented. The cycle is repeated until B = 0.

(Note that B is tested for zero after it is decremented. By loading B initially with zero, 256 data transfers will take place.)

(HL) ← (C)	S: Undefined
HL ← HL + 1	Z: Set
B ← B − 1	H: Undefined
Repeat until B = 0	P/V: Undefined
	N: Set
	C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	1	0	0	1	0

Timing: For B ≠ 0: M cycles — 5
T states — 21 (4, 5, 3, 4, 5)
For B = 0: M cycles — 4
T states — 16 (4, 5, 3, 4)

Addressing Mode: Implied/Source — Register Indirect
Destination — Register Indirect

(Note that at the end of each data transfer cycle, interrupts may be recognized and two refresh cycles will be performed.)

OTIR

Data is output to the I/O device at port address (C) from memory location (HL), the HL memory pointer is incremented, and the byte counter B is decremented. The cycles are repeated until B = 0.

(Note that B is tested for zero after it is decremented. By loading B initially with zero, 256 data transfers will take place.)

(C) ← (HL)	S: Undefined
HL ← HL + 1	H: Undefined
B ← B − 1	Z: Set
Repeat until B = 0	P/V: Undefined
	N: Set
	C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	1	0	0	1	1

Timing: For B ≠ 0: M cycles — 5
T states — 21 (4, 5, 3, 4, 5)
For B = 0: M cycles — 4
T states — 16 (4, 5, 3, 4)

Addressing Mode: Implied/Source — Register Indirect
Destination — Register Indirect

(Note that at the end of each data transfer cycle, interrupts may be recognized and two refresh cycles will be performed.)

12.12 Input/Output (Continued)

INDR

Data is input from the I/O device at address (C) to memory location (HL), then the HL memory pointer is byte counter B are decremented. The cycle is repeated until B = 0.

(Note that B is tested for zero after it is decremented. By loading B initially with zero, 256 data transfers will take place.)

(HL) ← (C)	S: Undefined
HL ← HL − 1	Z: Set
B ← B − 1	H: Undefined
Repeat until B = 0	P/V: Undefined
	N: Set
	C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	1	0	0	1	0

Timing:	For B ≠ 0	M cycles — 5
		T states — 21 (4, 5, 3, 4, 5)
	For B = 0	M cycles — 4
		T states — 16 (4, 5, 3, 4)
Addressing Mode:		Implied/Source — Register Indirect
		Destination — Register Indirect

(Note that after each data transfer cycle, interrupts may be recognized and two refresh cycles are performed.)

OTDR

Data is output from memory location (HL) to the I/O device at port address (C), then the HL memory pointer and byte counter B are decremented. The cycle is repeated until B = 0.

(Note that B is tested for zero after it is decremented. By loading B initially with zero, 256 data transfers will take place.)

(C) ← (HL)	S: Undefined
HL ← HL − 1	Z: Set
B ← B − 1	H: Undefined
Repeat until B = 0	P/V: Undefined
	N: Set
	C: N/A

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
1	0	1	1	1	0	1	1

Timing:	For B ≠ 0	M cycles — 5
		T states — 21 (4, 5, 3, 4, 5)
	For B = 0	M cycles — 4
		T states — 16 (4, 5, 3, 4)
Addressing Mode:		Implied/Source — Register Indirect
		Destination — Register Indirect

(Note that after each data transfer cycle the NSC800 will accept interrupts and perform two refresh cycles.)

12.13 CPU Control

NOP

The CPU performs no operation.

— — —   No flags affected

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

Timing:	M cycles — 1
	T states — 4
Addressing Mode:	N/A

HALT

The CPU halts execution of the program. Dummy op-code fetches are performed from the next memory location to keep the refresh circuits active until the CPU is interrupted or reset from the halted state.

— — —   No flags affected

7	6	5	4	3	2	1	0
0	1	1	1	0	1	1	0

Timing:	M cycles — 1
	T states — 4
Addressing Mode:	N/A

DI

Disable system level interrupts.

IFF_1 ← 0   No flags affected

IFF_2 ← 0

7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1

Timing:	M cycles — 1
	T states — 4
Addressing Mode:	N/A

EI

The system level interrupts are enabled. During execution of this instruction, and the next one, the maskable interrupts will be disabled.

IFF_1 ← 1   No flags affected

IFF_2 ← 1

7	6	5	4	3	2	1	0
1	1	1	1	1	0	1	1

Timing:	M cycles — 1
	T states — 4
Addressing Mode:	N/A

IM 0

The CPU is placed in interrupt mode 0.

— — —   No flags affected

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	0	0	0	1	1	0

Timing:	M cycles — 2
	T states — 8 (4, 4)
Addressing Mode:	N/A

12.13 CPU Control (Continued)

IM 1

The CPU is placed in interrupt mode 1.

— — — No flags affected

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	0	1	0	1	1	0

Timing: M cycles — 2
T states — 8 (4, 4)
Addressing Mode: N/A

IM 2

The CPU is placed in interrupt mode 2.

— — — No flags affected

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	0	1	1	1	1	0

Timing: M cycles — 2
T states — 8 (4, 4)
Addressing Mode: N/A

12.14 Program Control

JUMPS

JP nn

Unconditional jump to program location nn.

PC ← nn No flags affected

7	6	5	4	3	2	1	0
1	1	0	0	0	0	1	1
n (low-order byte)							
n (high-order byte)							

Timing: M cycles — 3
T states — 10 (4, 3, 3)
Addressing Mode: Direct

JP (ss)

Unconditional jump to program location pointed to by register ss.

PC ← ss No flags affected

7	6	5	4	3	2	1	0	
1	1	1	0	1	0	0	1	JP (HL)

Timing: M cycles — 1
T states — 4
Addressing Mode: Register Indirect

7	6	5	4	3	2	1	0	
1	1	N_X	1	1	1	0	1	JP (IX) (for N_X = 0) JP (IY) (for N_X = 1)
1	1	1	0	1	0	0	1	

Timing: M cycles — 2
T states — 8 (4, 4)
Addressing Mode: Register Indirect

JP cc, nn

Conditionally jump to program location nn based on testable flag states.

If cc true, No flags affected
PC ← nn,
otherwise continue

7	6	5	4	3	2	1	0
1	1	cc			0	1	0
n (low-order byte)							
n (high-order byte)							

Timing: M cycles — 3
T states — 10 (4, 3, 3)
Addressing Mode: Direct

JR d

Unconditional jump to program location calculated with respect to the program counter and the displacement d.

PC ← PC + d No flags affected

7	6	5	4	3	2	1	0
0	0	0	1	1	0	0	0
d − 2							

Timing: M cycles — 3
T states — 12 (4, 3, 5)
Addressing Mode: PC Relative

JR kk, d

Conditionally jump to program location calculated with respect to the program counter and the displacement d, based on limited testable flag states.

If kk true, No flags affected
PC ← PC + d,
otherwise continue

7	6	5	4	3	2	1	0
0	0	1	kk		0	0	0
d − 2							

Timing: if kk met (true) M cycles — 3
T states — 12 (4, 3, 5)
if kk not met (not true) M cycles — 2
T states — 7 (4, 3)
Addressing Mode: PC Relative

12.14 Program Control (Continued)

DJNZ d

Decrement the B register and conditionally jump to program location calculated with respect to the program counter and the displacement d, based on the contents of the B register.

B ← B − 1 — No flags affected

If B = 0 continue,

else PC ← PC + d

7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	0

d − 2

Timing: If B ≠ 0 — M cycles — 3; T states — 13 (5, 3, 5)

If B = 0 — M cycles — 2; T states — 8 (5, 3)

Addressing Mode: PC Relative

CALLS

CALL nn

Unconditional call to subroutine at location nn.

$(SP-1) \leftarrow PC_H$ — No flags affected

$(SP-2) \leftarrow PC_L$

SP ← SP − 2

PC ← nn

7	6	5	4	3	2	1	0
1	1	0	0	1	1	0	1

n (low-order byte)

n (high-order byte)

Timing: M Cycles — 5; T states — 17 (4, 3, 4, 3, 3)

Addressing Mode: Direct

CALL cc, nn

Conditional call to subroutine at location nn based on testable flag stages.

If cc true, — No flags affected

$(SP-1) \leftarrow PC_H$

$(SP-2) \leftarrow PC_L$

SP ← SP − 2

PC ← nn,

else continue

7	6	5	4	3	2	1	0
1	1	cc			1	0	0

n (low-order byte)

n (high-order byte)

Timing: If cc true — M cycles — 5; T states 17 (4, 3, 4, 3, 3)

If cc not true — M cycles — 3; T states — 10 (4, 3, 3)

Addressing Mode: Direct

RETURNS

RET

Unconditional return from subroutine or other return to program location pointed to by the top of the stack.

$PC_L \leftarrow (SP)$ — No flags affected

$PC_H \leftarrow (SP+1)$

SP ← SP + 2

7	6	5	4	3	2	1	0
1	1	0	0	1	0	0	1

Timing: M cycles — 3; T states — 10 (4, 3, 3)

Addressing Mode: Register Indirect

RET cc

Conditional return from subroutine or other return to program location pointed to by the top of the stack.

If cc true, — No flags affected

$PC_L \leftarrow (SP)$

$PC_H \leftarrow (SP+1)$

SP ← SP + 2,

else continue

7	6	5	4	3	2	1	0
1	1	cc			0	0	0

Timing: If cc true — M cycles — 3; T states — 11 (5, 3, 3)

If cc not true — M cycles — 1; T states — 5

Addressing Mode: Register Indirect

RETI

Unconditional return from interrupt handling subroutine. Functionally identical to RET instruction. Unique opcode allows monitoring by external hardware.

$PC_L \leftarrow (SP)$ — No flags affected

$PC_H \leftarrow (SP+1)$

SP ← SP + 2

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	0	0	1	1	0	1

Timing: M cycles — 4; T states — 14 (4, 4, 3, 3)

Addressing Mode: Register Indirect

12.14 Program Control (Continued)

RETN

Unconditional return from non-maskable interrupt handling subroutine. Functionally similar to RET instruction, except interrupt enable state is restored to that prior to non-maskable interrupt.

$PC_L \leftarrow (SP)$ No flags affected

$PC_H \leftarrow (SP + 1)$

$SP \leftarrow SP + 2$

$IFF_1 \leftarrow IFF_2$

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1
0	1	0	0	0	1	0	1

Timing: M cycles — 4

T states — 14 (4, 4, 3, 3)

Addressing Mode: Register Indirect

RESTARTS

RST P

The present contents of the PC are pushed onto the memory stack and the PC is loaded with dedicated program locations as determined by the specific restart executed.

$(SP - 1) \leftarrow PC_H$ No flags affected

$(SP - 2) \leftarrow PC_L$

$SP \leftarrow SP - 2$

$PC_H \leftarrow 0$

$PC_L \leftarrow P$

7	6	5	4	3	2	1	0
1	1		t		1	1	1

Timing: M cycles — 3

T states — 11 (5, 3, 3)

Addressing Mode: Modified Page Zero

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

12.15 Instruction Set: Alphabetical Order

ADC	A, (HL)	8E	BIT	0, B	CB 40
ADC	A, (IX+d)	DD 8Ed	BIT	0, C	CB 41
ADC	A, (IY+d)	FD 8Ed	BIT	0, D	CB 42
ADC	A, A	8F	BIT	0, E	CB 43
ADC	A, B	88	BIT	0, H	CB 44
ADC	A, C	89	BIT	0, L	CB 45
ADC	A, D	8A	BIT	1, (HL)	CB 4E
ADC	A, E	8B	BIT	1, (IX+d)	DD CBd4E
ADC	A, H	8C	BIT	1, (IY+d)	FD CBd4E
ADC	A, L	8D	BIT	1, A	CB 4F
ADC	A, n	CE n	BIT	1, B	CB 48
ADC	HL, BC	ED 4A	BIT	1, C	CB 49
ADC	HL, DE	ED 5A	BIT	1, D	CB 4A
ADC	HL, HL	ED 6A	BIT	1, E	CB 4B
ADC	HL, SP	ED 7A	BIT	1, H	CB 4C
ADD	A, (HL)	86	BIT	1, L	CB 4D
ADD	A, (IX+d)	DD 86d	BIT	2, (HL)	CB 56
ADD	A, (IY+d)	FD 86d	BIT	2, (IX+d)	DD CBd56
ADD	A, A	87	BIT	2, (IY+d)	FD CBd56
ADD	A, B	80	BIT	2, A	CB 57
ADD	A, C	81	BIT	2, B	CB 50
ADD	A, D	82	BIT	2, C	CB 51
ADD	A, E	83	BIT	2, D	CB 52
ADD	A, H	84	BIT	2, E	CB 53
ADD	A, L	85	BIT	2, H	CB 54
ADD	A, n	C6 n	BIT	2, L	CB 55
ADD	HL, BC	09	BIT	3, (HL)	CB 5E
ADD	HL, DE	19	BIT	3, (IX+d)	DD CBd5E
ADD	HL, HL	29	BIT	3, (IY+d)	FD CBd5E
ADD	HL, SP	39	BIT	3, A	CB 5F
ADD	IX, BC	DD 09	BIT	3, B	CB 58
ADD	IX, DE	DD 19	BIT	3, C	CB 59
ADD	IX, IX	DD 29	BIT	3, D	CB 5A
ADD	IX, SP	DD 39	BIT	3, E	CB 5B
ADD	IY, BC	FD 09	BIT	3, H	CB 5C
ADD	IY, DE	FD 19	BIT	3, L	CB 5D
ADD	IY, IY	FD 29	BIT	4, (HL)	CB 66
ADD	IY, SP	FD 39	BIT	4, (IX+d)	DD CBd66
AND	(HL)	A6	BIT	4, (IY+d)	FD CBd66
AND	(IX+d)	DD A6d	BIT	4, A	CB 67
AND	(IY+d)	FD A6d	BIT	4, B	CB 60
AND	A	A7	BIT	4, C	CB 61
AND	B	A0	BIT	4, D	CB 62
AND	C	A1	BIT	4, E	CB 63
AND	D	A2	BIT	4, H	CB 64
AND	E	A3	BIT	4, L	CB 65
AND	H	A4	BIT	5, (HL)	CB 6E
AND	L	A5	BIT	5, (IX+d)	DD CBd6E
AND	n	E6 n	BIT	5, (IY+d)	FD CBd6E
BIT	0, (HL)	CB 46	BIT	5, A	CB 6F
BIT	0, (IX+d)	DD CBd46	BIT	5, B	CB 68
BIT	0, (IY+d)	FD CBd46	BIT	5, C	CB 69
BIT	0, A	CB 47	BIT	5, D	CB 6A

(nn) = address of memory location   d = signed displacement
nn = Data (16 bit)   d2 = d − 2
n = Data (8 bit)

12.15 Instruction Set: Alphabetical Order (Continued)

BIT	5, E	CB 6B
BIT	5, H	CB 6C
BIT	5, L	CB 6D
BIT	6, (HL)	CB 76
BIT	6, (IX+d)	DD CBd76
BIT	6, (IY+d)	FD CBd76
BIT	6, A	CB 77
BIT	6, B	CB 70
BIT	6, C	CB 71
BIT	6, D	CB 72
BIT	6, E	CB 73
BIT	6, H	CB 74
BIT	6, L	CB 75
BIT	7, (HL)	CB 7E
BIT	7, (IX+d)	DD CBd7E
BIT	7, (IY+d)	FD CBd7E
BIT	7, A	CB 7F
BIT	7, B	CB 78
BIT	7, C	CB 79
BIT	7, D	CB 7A
BIT	7, E	CB 7B
BIT	7, H	CB 7C
BIT	7, L	CB 7D
CALL	C, nn	DCnn
CALL	M, nn	FCnn
CALL	NC, nn	D4nn
CALL	nn	CDnn
CALL	NZ, nn	C4nn
CALL	P, nn	F4nn
CALL	PE, nn	ECnn
CALL	PO, nn	E4nn
CALL	Z, nn	CCnn
CCF		3F
CP	(HL)	BE
CP	(IX+d)	DD BEd
CP	(IY+d)	FD BEd
CP	A	BF
CP	B	B8
CP	C	B9
CP	D	BA
CP	E	BB
CP	H	BC
CP	L	BD
CP	n	FE n
CPD		ED A9
CPDR		ED B9
CPI		ED A1
CPIR		ED B1
CPL		2F
DAA		27
DEC	(HL)	35
DEC	(IX+d)	DD 35d
DEC	(IY+d)	FD 35d

(nn) = Address of memory location
nn = Data (16 bit)
n = Data (8 bit)
d = signed displacement
d2 = d − 2

DEC	A	3D
DEC	B	05
DEC	BC	0B
DEC	C	0D
DEC	D	15
DEC	DE	1B
DEC	E	1D
DEC	H	25
DEC	HL	2B
DEC	IX	DD 2B
DEC	IY	FD 2B
DEC	L	2D
DEC	SP	3B
DI		F3
DJNZ	d2	10 d2
EI		FB
EX	(SP), HL	E3
EX	(SP), IX	DD E3
EX	(SP), IY	FD E3
EX	AF, A'F'	08
EX	DE, HL	EB
EXX		D9
HALT		76
IM	0	ED 46
IM	1	ED 56
IM	2	ED 5E
IN	A, (C)	ED78
IN	A, (n)	DB n
IN	B, (C)	ED 40
IN	C, (C)	ED 48
IN	D, (C)	ED 50
IN	E, (C)	ED 58
IN	H, (C)	ED 60
IN	L, (C)	ED 68
INC	(HL)	34
INC	(IX+d)	DD 34d
INC	(IY+d)	FD 34d
INC	A	3C
INC	B	04
INC	BC	03
INC	C	0C
INC	D	14
INC	DE	13
INC	E	1C
INC	H	24
INC	HL	23
INC	IX	DD 23
INC	IY	FD 23
INC	L	2C
INC	SP	33
IND		ED AA
INDR		ED BA
INI		ED A2

12.15 Instruction Set: Alphabetical Order (Continued)

INIR		ED B2
JP	(HL)	E9
JP	(IX)	DD E9
JP	(IY)	FD E9
JP	C, nn	DAnn
JP	M, nn	FAnn
JP	NC, nn	D2nn
JP	nn	C3nn
JP	NZ, nn	C2nn
JP	P, nn	F2nn
JP	PE, nn	EAnn
JP	PO, nn	E2nn
JP	Z, nn	CAnn
JR	C, d2	38 d2
JR	d2	18 d2
JR	NC, d2	30 d2
JR	NZ, d2	20 d2
JR	Z, d2	28 d2
LD	(BC), A	02
LD	(DE), A	12
LD	(HL), A	77
LD	(HL), B	70
LD	(HL), C	71
LD	(HL), D	72
LD	(HL), E	73
LD	(HL), H	74
LD	(HL), L	75
LD	(HL), n	36 n
LD	(IX+d), A	DD 77d
LD	(IX+d), B	DD 70d
LD	(IX+d), C	DD 71d
LD	(IX+d), D	DD 72d
LD	(IX+d), E	DD 73d
LD	(IX+d), H	DD 74d
LD	(IX+d), L	DD 75d
LD	(IX+d), n	DD 36dn
LD	(IY+d), A	FD 77d
LD	(IY+d), B	FD 70d
LD	(IY+d), C	FD 71d
LD	(IY+d), D	FD 72d
LD	(IY+d), E	FD 73d
LD	(IY+d), H	FD 74d
LD	(IY+d), L	FD 75d
LD	(IY+d), n	FD 36dn
LD	(nn), A	32nn
LD	(nn), BC	ED 43nn
LD	(nn), DE	ED 53nn
LD	(nn), HL	22nn
LD	(nn), IX	DD 22nn
LD	(nn), IY	FD 22nn
LD	(nn), SP	ED 73nn
LD	A, (BC)	0A
LD	A, (DE)	1A

LD	A, (HL)	7E
LD	A, (IX+d)	DD 7Ed
LD	A, (IY+d)	FD 7Ed
LD	A, (nn)	3Ann
LD	A, A	7F
LD	A, B	78
LD	A, C	79
LD	A, D	7A
LD	A, E	7B
LD	A, H	7C
LD	A, I	ED 57
LD	A, L	7D
LD	A, n	3E n
LD	B, (HL)	46
LD	B, (IX+d)	DD 46d
LD	B, (IY+d)	FD 46d
LD	B, A	47
LD	B, B	40
LD	B, C	41
LD	B, D	42
LD	B, E	43
LD	B, H	44
LD	B, L	45
LD	B, n	06 n
LD	BC, (nn)	ED 4B
LD	BC, nn	01nn
LD	C, (HL)	4E
LD	C, (IX+d)	DD 4Ed
LD	C, (IY+d)	FD 4Ed
LD	C, A	4F
LD	C, B	48
LD	C, C	49
LD	C, D	4A
LD	C, E	4B
LD	C, H	4C
LD	C, L	4D
LD	C, n	0E n
LD	D, (HL)	56
LD	D, (IX+d)	DD 56d
LD	D, (IY+d)	FD 56d
LD	D, A	57
LD	D, B	50
LD	D, C	51
LD	D, D	52
LD	D, E	53
LD	D, H	54
LD	D, L	55
LD	D, n	16 n
LD	DE, (nn)	ED 5Bnn
LD	DE, nn	11nn
LD	E, (HL)	5E
LD	E, (IX+d)	DD 5Ed
LD	E, (IY+d)	FD 5Ed

(nn) = Address of memory location
nn = Data (16 bit)
n = Data (8 bit)
d = signed displacement
d2 = d − 2

12.15 Instruction Set: Alphabetical Order (Continued)

LD	E, A	5F
LD	E, B	58
LD	E, C	59
LD	E, D	5A
LD	E, E	5B
LD	E, H	5C
LD	E, L	5D
LD	E, n	1E n
LD	H, (HL)	66
LD	H, (IX+d)	DD 66d
LD	H, (IY+d)	FD 66d
LD	H, A	67
LD	H, B	60
LD	H, C	61
LD	H, D	62
LD	H, E	63
LD	H, H	64
LD	H, L	65
LD	H, n	26 n
LD	HL, (nn)	2Ann
LD	HL, nn	21nn
LD	I, A	ED 47
LD	IX, (nn)	DD 2Ann
LD	IX, nn	DD 21nn
LD	IY, (nn)	FD 2Ann
LD	IY, nn	FD 21nn
LD	L, (HL)	6E
LD	L, (IX+d)	DD 6Ed
LD	L, (IY+d)	FD 6Ed
LD	L, A	6F
LD	L, B	68
LD	L, C	69
LD	L, D	6A
LD	L, E	6B
LD	L, H	6C
LD	L, L	6D
LD	L, n	2E n
LD	SP, (nn)	ED 7Bnn
LD	SP, HL	F9
LD	SP, IX	DD F9
LD	SP, IY	FD F9
LD	SP, nn	31nn
LDD		ED A8
LDDR		ED B8
LDI		ED A0
LDIR		ED B0
NEG		ED n
NOP		00
OR	(HL)	B6
OR	(IX+d)	DD B6d
OR	(IY+d)	FD B6d
OR	A	B7
OR	B	B0
OR	C	B1
OR	D	B2
OR	E	B3
OR	H	B4
OR	L	B5
OR	n	F6 n
OTDR		ED BB
OTIR		ED B3
OUT	(C), A	ED 79
OUT	(C), B	ED 41
OUT	(C), C	ED 49
OUT	(C), D	ED 51
OUT	(C), E	ED 59
OUT	(C), H	ED 61
OUT	(C), L	ED 69
OUT	n, A	D3 n
OUTD		ED AB
OUTI		ED A3
POP	AF	F1
POP	BC	C1
POP	DE	D1
POP	HL	E1
POP	IX	DD E1
POP	IY	FD E1
PUSH	AF	F5
PUSH	BC	C5
PUSH	DE	D5
PUSH	HL	E5
PUSH	IX	DD E5
PUSH	IY	FD E5
RES	0, (HL)	CB 86
RES	0, (IX+d)	DD CBd86
RES	0, (IY+d)	FD CBd86
RES	0, A	CB 87
RES	0, B	CB 80
RES	0, C	CB 81
RES	0, D	CB 82
RES	0, E	CB 83
RES	0, H	CB 84
RES	0, L	CB 85
RES	1, (HL)	CB 8E
RES	1, (IX+d)	DD CBd8E
RES	1, (IY+d)	FD CBd8E
RES	1, A	CB 8F
RES	1, B	CB 88
RES	1, C	CB 89
RES	1, D	CB 8A
RES	1, E	CB 8B
RES	1, H	CB 8C
RES	1, L	CB 8D
RES	2, (HL)	CB 96
RES	2, (IX+d)	DD CBd96
RES	2, (IY+d)	FD CBd96

(nn) = Address of memory location
nn = Data (16 bit)
n = Data (8 bit)
d = signed displacement
d2 = d − 2

12.15 Instruction Set: Alphabetical Order (Continued)

RES	2, A	CB 97
RES	2, B	CB 90
RES	2, C	CB 91
RES	2, D	CB 92
RES	2, E	CB 93
RES	2, H	CB 94
RES	2, L	CB 95
RES	3, (HL)	CB 9E
RES	3, (IX+d)	DD CBd9E
RES	3, (IY+d)	FD CBd9E
RES	3, A	CB 9F
RES	3, B	CB 98
RES	3, C	CB 99
RES	3, D	CB 9A
RES	3, E	CB 9B
RES	3, H	CB 9C
RES	3, L	CB 9D
RES	4, (HL)	CB A6
RES	4, (IX+d)	DD CBdA6
RES	4, (IY+d)	FD CBdA6
RES	4, A	CB A7
RES	4, B	CB A0
RES	4, C	CB A1
RES	4, D	CB A2
RES	4, E	CB A3
RES	4, H	CB A4
RES	4, L	CB A5
RES	5, (HL)	CB AE
RES	5, (IX+d)	DD CBdAE
RES	5, (IY+d)	FD CBdAE
RES	5, A	CB AF
RES	5, B	CB A8
RES	5, C	CB A9
RES	5, D	CB AA
RES	5, E	CB AB
RES	5, H	CB AC
RES	5, L	CB AD
RES	6, (HL)	CB B6
RES	6, (IX+d)	DD CBdB6
RES	6, (IY+d)	FD CBdB6
RES	6, A	CB B7
RES	6, B	CB B0
RES	6, C	CB B1
RES	6, D	CB B2
RES	6, E	CB B3
RES	6, H	CB B4
RES	6, L	CB B5
RES	7, (HL)	CB BE
RES	7, (IX+d)	DD CBdBE
RES	7, (IY+d)	FD CBdBE
RES	7, A	CB BF
RES	7, B	CB B8
RES	7, C	CB B9
RES	7, D	CB BA
RES	7, E	CB BB
RES	7, H	CB BC
RES	7, L	CB BD
RET		C9
RET	C	D8
RET	M	F8
RET	NC	D0
RET	NZ	C0
RET	P	F0
RET	PE	E8
RET	PO	E0
RET	Z	C8
RETI		ED 4D
RETN		ED 45
RL	(HL)	CB 16
RL	(IX+d)	DD CBd16
RL	(IY+d)	FD CBd16
RL	A	CB 17
RL	B	CB 10
RL	C	CB 11
RL	D	CB 12
RL	E	CB 13
RL	H	CB 14
RL	L	CB 15
RLA		17
RLC	(HL)	CB 06
RLC	(IX+d)	DD CBd06
RLC	(IY+d)	FD CBd06
RLC	A	CB 07
RLC	B	CB 00
RLC	C	CB 01
RLC	D	CB 02
RLC	E	CB 03
RLC	H	CB 04
RLC	L	CB 05
RLCA		07
RLD		ED 6F
RR	(HL)	CB 1E
RR	(IX+d)	DD CBd1E
RR	(IY+d)	FD CBd1E
RR	A	CB 1F
RR	B	CB 18
RR	C	CB 19
RR	D	CB 1A
RR	E	CB 1B
RR	H	CB 1C
RR	L	CB 1D
RRA		1F
RRC	(HL)	CB 0E
RRC	(IX+d)	DD CBd0E
RRC	(IY+d)	FD CBd0E
RRC	A	CB 0F

(nn) = Address of memory location
nn = Data (16 bit)
n = Data (8 bit)
d = signed displacement
d2 = d − 2

12.15 Instruction Set: Alphabetical Order (Continued)

RRC	B	CB 08
RRC	C	CB 09
RRC	D	CB 0A
RRC	E	CB 0B
RRC	H	CB 0C
RRC	L	CB 0D
RRCA		0F
RRD		ED 67
RST	0	C7
RST	08H	CF
RST	10H	D7
RST	18H	DF
RST	20H	E7
RST	28H	EF
RST	30H	F7
RST	38H	FF
SBC	A, (HL)	9E
SBC	A, (IX+d)	DD 9Ed
SBC	A, (IY+d)	FD 9Ed
SBC	A, A	9F
SBC	A, B	98
SBC	A, C	99
SBC	A, D	9A
SBC	A, E	9B
SBC	A, H	9C
SBC	A, L	9D
SBC	A, n	DE n
SBC	HL, BC	ED 42
SBC	HL, DE	ED 52
SBC	HL, HL	ED 62
SBC	HL, SP	ED 72
SCF		37
SET	0, (HL)	CB C6
SET	0, (IX+d)	DD CBdC6
SET	0, (IY+d)	FD CBdC6
SET	0, A	CB C7
SET	0, B	CB C0
SET	0, C	CB C1
SET	0, D	CB C2
SET	0, E	CB C3
SET	0, H	CB C4
SET	0, L	CB C5
SET	1, (HL)	CB CE
SET	1, (IX+d)	DD CBdCE
SET	1, (IY+d)	FD CBdCE
SET	1, A	CB CF
SET	1, B	CB C8
SET	1, C	CB C9
SET	1, D	CB CA
SET	1, E	CB CB
SET	1, H	CB CC
SET	1, L	CB CD
SET	2, (HL)	CB D6
SET	2, (IX+d)	DD CBdD6
SET	2, (IY+d)	FD CBdD6
SET	2, A	CB D7
SET	2, B	CB D0
SET	2, C	CB D1
SET	2, D	CB D2
SET	2, E	CB D3
SET	2, H	CB D4
SET	2, L	CB D5
SET	3, (HL)	CB DE
SET	3, (IX+d)	DD CBdDE
SET	3, (IY+d)	FD CBdDE
SET	3, A	CB DF
SET	3, B	CB D8
SET	3, C	CB D9
SET	3, D	CB DA
SET	3, E	CB DB
SET	3, H	CB DC
SET	3, L	CB DD
SET	4, (HL)	CB E6
SET	4, (IX+d)	DD CBdE6
SET	4, (IY+d)	FD CBdE6
SET	4, A	CB E7
SET	4, B	CB E0
SET	4, C	CB E1
SET	4, D	CB E2
SET	4, E	CB E3
SET	4, H	CB E4
SET	4, L	CB E5
SET	5, (HL)	CB EE
SET	5, (IX+d)	DD CBdEE
SET	5, (IY+d)	FD CBdEE
SET	5, A	CB EF
SET	5, B	CB E8
SET	5, C	CB E9
SET	5, D	CB EA
SET	5, E	CB EB
SET	5, H	CB EC
SET	5, L	CB ED
SET	6, (HL)	CB F6
SET	6, (IX+d)	DD CBdF6
SET	6, (IY+d)	FD CBdF6
SET	6, A	CB F7
SET	6, B	CB F0
SET	6, C	CB F1
SET	6, D	CB F2
SET	6, E	CB F3
SET	6, H	CB F4
SET	6, L	CB F5
SET	7, (HL)	CB FE
SET	7, (IX+d)	DD CBdFE
SET	7, (IY+d)	FD CBdFE
SET	7, A	CB FF

(nn) = Address of memory location
nn = Data (16 bit)
n = Data (8 bit)
d = displacement
d2 = d − 2

12.15 Instruction Set: Alphabetical Order (Continued)

SET	7, B	CB F8
SET	7, C	CB F9
SET	7, D	CB FA
SET	7, E	CB FB
SET	7, H	CB FC
SET	7, L	CB FD
SLA	(HL)	CB 26
SLA	(IX+d)	DD CBd26
SLA	(IY+d)	FD CBd26
SLA	A	CB 27
SLA	B	CB 20
SLA	C	CB 21
SLA	D	CB 22
SLA	E	CB 23
SLA	H	CB 24
SLA	L	CB 25
SRA	(HL)	CB 2E
SRA	(IX+d)	DD CBd2E
SRA	(IY+d)	FD CBd2E
SRA	A	CB 2F
SRA	B	CB 28
SRA	C	CB 29
SRA	D	CB 2A
SRA	E	CB 2B
SRA	H	CB 2C
SRA	L	CB 2D
SRL	(HL)	CB 3E
SRL	(IX+d)	DD CBd3E
SRL	(IY+d)	FD CBd3E

SRL	A	CB 3F
SRL	B	CB 38
SRL	C	CB 39
SRL	D	CB 3A
SRL	E	CB 3B
SRL	H	CB 3C
SRL	L	CB 3D
SUB	(HL)	96
SUB	(IX+d)	DD 96d
SUB	(IY+d)	FD 96d
SUB	A	97
SUB	B	90
SUB	C	91
SUB	D	92
SUB	E	93
SUB	H	94
SUB	L	95
SUB	n	D6 n
XOR	(HL)	AE
XOR	(IX+d)	DD AEd
XOR	(IY+d)	FD AEd
XOR	A	AF
XOR	B	A8
XOR	C	A9
XOR	D	AA
XOR	E	AB
XOR	H	AC
XOR	L	AD
XOR	n	EE n

12.16 Instruction Set: Numerical Order

Op Code	Mnemonic
00	NOP
01nn	LD BC,nn
02	LD (BC),A
03	INC BC
04	INC B
05	DEC B
06n	LD B,n
07	RLCA
08	EX AF,A'F'
09	ADD HL,BC
0A	LD A,(BC)
0B	DEC BC
0C	INC C
0D	DEC C
0En	LD C,n
0F	RRCA
10d2	DJNZ d2
11nn	LD DE,nn
12	LD (DE),A
13	INC DE
14	INC D

Op Code	Mnemonic
15	DEC D
16n	LD D,n
17	RLA
18d2	JR d2
19	ADD HL,DE
1A	LD A,(DE)
1B	DEC DE
1C	INC E
1D	DEC E
1En	LD E,n
1F	RRA
20d2	JR NZ,d2
21nn	LD HL,nn
22nn	LD (nn),HL
23	INC HL
24	INC H
25	DEC H
26n	LD H, n
27	DAA
28d2	JR Z,d2
29	ADD HL,HL

Op Code	Mnemonic
2Ann	LD HL,(nn)
2B	DEC HL
2C	INC L
2D	DEC L
2En	LD L,n
2F	CPL
30d2	JR NC,d2
31nn	LD SP,nn
32nn	LD (nn),A
33	INC SP
34	INC (HL)
35	DEC (HL)
36n	LD (HL),n
37	SCF
38	JR C,d2
39	ADD HL,SP
3Ann	LD A,(nn)
3B	DEC SP
3C	INC A
3D	DEC A
3En	LD A,n

(nn) = Address of memory location
nn = Data (16 bit)
n = Data (8 bit)
d = displacement
d2 = d − 2

12.16 Instruction Set: Numerical Order (Continued)

Op Code	Mnemonic
3F	CCF
40	LD B,B
41	LD B,C
42	LD B,D
43	LD B,E
44	LD B,H
45	LD B,L
46	LD B,(HL)
47	LD B,A
48	LD C,B
49	LD C,C
4A	LD C,D
4B	LD C,E
4C	LD C,H
4D	LD C,L
4E	LD C,(HL)
4F	LD C,A
50	LD D,B
51	LD D,C
52	LD D,D
53	LD D,E
54	LD D,H
55	LD D,L
56	LD D,(HL)
57	LD D,A
58	LD E,B
59	LD E,C
5A	LD E,D
5B	LD E,E
5C	LD E,H
5D	LD E,L
5E	LD E,(HL)
5F	LD E,A
60	LD H,B
61	LD H,C
62	LD H,D
63	LD H,E
64	LD H,H
65	LD H,L
66	LD H,(HL)
67	LD H,A
68	LD L,B
69	LD L,C
6A	LD L,D
6B	LD L,E
6C	LD L,H
6D	LD L,L
6E	LD L,(HL)
6F	LD L,A
70	LD (HL),B
71	LD (HL),C
72	LD (HL),D
73	LD (HL),E

Op Code	Mnemonic
74	LD (HL),H
75	LD (HL),L
76	HALT
77	LD (HL),A
78	LD A,B
79	LD A,C
7A	LD A,D
7B	LD A,E
7C	LD A,H
7D	LD A,L
7E	LD A,(HL)
7F	LD A,A
80	ADD A,B
81	ADD A,C
82	ADD A,D
83	ADD A,E
84	ADD A,H
85	ADD A,L
86	ADD A,(HL)
87	ADD A,A
88	ADC A,B
89	ADC A,C
8A	ADC A,D
8B	ADC A,E
8C	ADC A,H
8D	ADC A,L
8E	ADC A,(HL)
8F	ADC A,A
90	SUB B
91	SUB C
92	SUB D
93	SUB E
94	SUB H
95	SUB L
96	SUB (HL)
97	SUB A
98	SBC A,B
99	SBC A,C
9A	SBC A,D
9B	SBC A,E
9C	SBC A,H
9D	SBC A,L
9E	SBC A,(HL)
9F	SBC A,A
A0	AND B
A1	AND C
A2	AND D
A3	AND E
A4	AND H
A5	AND L
A6	AND (HL)
A7	AND A
A8	XOR B

Op Code	Mnemonic
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
AE	XOR (HL)
AF	XOR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E
B4	OR H
B5	OR L
B6	OR (HL)
B7	OR A
B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H
BD	CP L
BE	CP (HL)
BF	CP A
C0	RET NZ
C1	POP BC
C2nn	JP NZ,nn
C3nn	JP nn
C4nn	CALL NZ,nn
C5	PUSH BC
C6n	ADD A,n
C7	RST 0
C8	RET Z
C9	RET
CAnn	JP Z,nn
CB00	RLC B
CB01	RLC C
CB02	RLC D
CB03	RLC E
CB04	RLC H
CB05	RLC L
CB06	RLC (HL)
CB07	RLC A
CB08	RRC B
CB09	RRC C
CB0A	RRC D
CB0B	RRC E
CB0C	RRC H
CB0D	RRC L
CB0E	RRC (HL)
CB0F	RRC A
CB10	RL B
CB11	RL C
CB12	RL D

(nn) = Address of memory location d = displacement
nn = Data (16 bit) d2 = d − 2
n = Data (8-bit)

12.16 Instruction Set: Numerical Order (Continued)

Op Code	Mnemonic	Op Code	Mnemonic	Op Code	Mnemonic
CB13	RL E	CB4F	BIT 1,A	CB83	RES 0,E
CB14	RL H	CB50	BIT 2,B	CB84	RES 0,H
CB15	RL L	CB51	BIT 2,C	CB85	RES 0,L
CB16	RL (HL)	CB52	BIT 2,D	CB86	RES 0,(HL)
CB17	RL A	CB53	BIT 2,E	CB87	RES 0,A
CB18	RR B	CB54	BIT 2,H	CB88	RES 1,B
CB19	RR C	CB55	BIT 2,L	CB89	RES 1,C
CB1A	RR D	CB56	BIT 2,(HL)	CB8A	RES 1,D
CB1B	RR E	CB57	BIT 2,A	CB8B	RES 1,E
CB1C	RR H	CB58	BIT 3,B	CB8C	RES 1,H
CB1D	RR L	CB59	BIT 3,C	CB8D	RES 1,L
CB1E	RR (HL)	CB5A	BIT 3,D	CB8E	RES 1,(HL)
CB1F	RR A	CB5B	BIT 3,E	CB8F	RES 1,A
CB20	SLA B	CB5C	BIT 3,H	CB90	RES 2,B
CB21	SLA C	CB5D	BIT 3,L	CB91	RES 2,C
CB22	SLA D	CB5E	BIT 3,(HL)	CB92	RES 2,D
CB23	SLA E	CB5F	BIT 3,A	CB93	RES 2,E
CB24	SLA H	CB60	BIT 4,B	CB94	RES 2,H
CB25	SLA L	CB61	BIT 4,C	CB95	RES 2,L
CB26	SLA (HL)	CB62	BIT 4,D	CB96	RES 2,(HL)
CB27	SLA A	CB63	BIT 4,E	CB97	RES 2,A
CB28	SRA B	CB64	BIT 4,H	CB98	RES 3,B
CB29	SRA C	CB65	BIT 4,L	CB99	RES 3,C
CB2A	SRA D	CB66	BIT 4,(HL)	CB9A	RES 3,D
CB2B	SRA E	CB67	BIT 4,A	CB9B	RES 3,E
CB2C	SRA H	CB68	BIT 5,B	CB9C	RES 3,H
CB2D	SRA L	CB69	BIT 5,C	CB9D	RES 3,L
CB2E	SRA (HL)	CB6A	BIT 5,D	CB9E	RES 3,(HL)
CB2F	SRA A	CB6B	BIT 5,E	CB9F	RES 3,A
CB38	SRL B	CB6C	BIT 5,H	CBA0	RES 4,B
CB39	SRL C	CB6D	BIT 5,L	CBA1	RES 4,C
CB3A	SRL D	CB6E	BIT 5,(HL)	CBA2	RES 4,D
CB3B	SRL E	CB6F	BIT 5,A	CBA3	RES 4,E
CB3C	SRL H	CB70	BIT 6,B	CBA4	RES 4,H
CB3D	SRL L	CB71	BIT 6,C	CBA5	RES 4,L
CB3E	SRL (HL)	CB72	BIT 6,D	CBA6	RES 4,(HL)
CB3F	SRL A	CB73	BIT 6,E	CBA7	RES 4,A
CB40	BIT 0,B	CB74	BIT 6,H	CBA8	RES 5,B
CB41	BIT 0,C	CB75	BIT 6,L	CBA9	RES 5,C
CB42	BIT 0,D	CB76	BIT 6,(HL)	CBAA	RES 5,D
CB43	BIT 0,E	CB77	BIT 6,A	CBAB	RES 5,E
CB44	BIT 0,H	CB78	BIT 7,B	CBAC	RES 5,H
CB45	BIT 0,L	CB79	BIT 7,C	CBAD	RES 5,L
CB46	BIT 0,(HL)	CB7A	BIT 7,D	CBAE	RES 5,(HL)
CB47	BIT 0,A	CB7B	BIT 7,E	CBAF	RES 5,A
CB48	BIT 1,B	CB7C	BIT 7,H	CBB0	RES 6,B
CB49	BIT 1,C	CB7D	BIT 7,L	CBB1	RES 6,C
CB4A	BIT 1,D	CB7E	BIT 7,(HL)	CBB2	RES 6,D
CB4B	BIT 1,E	CB7F	BIT 7,A	CBB3	RES 6,E
CB4C	BIT 1,H	CB80	RES 0,B	CBB4	RES 6,H
CB4D	BIT 1,L	CB81	RES 0,C	CBB5	RES 6,L
CB4E	BIT 1,(HL)	CB82	RES 0,D	CBB6	RES 6,(HL)

(nn) = Address of memory location d = displacement
nn = Data (16 bit) d2 = d − 2
n = Data (8-bit)

12.16 Instruction Set: Numerical Order (Continued)

Op Code	Mnemonic
CBB7	RES 6,A
CBB8	RES 7,B
CBB9	RES 7,C
CBBA	RES 7,D
CBBB	RES 7,E
CBBC	RES 7,H
CBBD	RES 7,L
CBBE	RES 7,(HL)
CBBF	RES 7,A
CBC0	SET 0,B
CBC1	SET 0,C
CBC2	SET 0,D
CBC3	SET 0,E
CBC4	SET 0,H
CBC5	SET 0,L
CBC6	SET 0,(HL)
CBC7	SET 0,A
CBC8	SET 1,B
CBC9	SET 1,C
CBCA	SET 1,D
CBCB	SET 1,E
CBCC	SET 1,H
CBCD	SET 1,L
CBCE	SET 1,(HL)
CBCF	SET 1,A
CBD0	SET 2,B
CBD1	SET 2,C
CBD2	SET 2,D
CBD3	SET 2,E
CBD4	SET 2,H
CBD5	SET 2,L
CBD6	SET 2,(HL)
CBD7	SET 2,A
CBD8	SET 3,B
CBD9	SET 3,C
CBDA	SET 3,D
CBDB	SET 3,E
CBDC	SET 3,H
CBDD	SET 3,L
CBDE	SET 3,(HL)
CBDF	SET 3,A
CBE0	SET 4,B
CBE1	SET 4,C
CBE2	SET 4,D
CBE3	SET 4,E
CBE4	SET 4,H
CBE5	SET 4,L
CBE6	SET 4,(HL)
CBE7	SET 4,A
CBE8	SET 5,B
CBE9	SET 5,C
CBEA	SET 5,D
CBEB	SET 5,E
CBEC	SET 5,H
CBED	SET 5,L
CBEE	SET 5,(HL)
CBEF	SET 5,A
CBF0	SET 6,B
CBF1	SET 6,C
CBF2	SET 6,D
CBF3	SET 6,E
CBF4	SET 6,H
CBF5	SET 6,L
CBF6	SET 6,(HL)
CBF7	SET 6,A
CBF8	SET 7,B
CBF9	SET 7,C
CBFA	SET 7,D
CBFB	SET 7,E
CBFC	SET 7,H
CBFD	SET 7,L
CBFE	SET 7,(HL)
CBFF	SET 7,A
CCnn	CALL Z,nn
CDnn	CALL nn
CEn	ADC A,n
CF	RST 8
D0	RET NC
D1	POP DE
D2nn	JP NC,nn
D3n	OUT (n),A
D4nn	CALL NC,nn
D5	PUSH DE
D6n	SUB n
D7	RST 10H
D8	RET C
D9	EXX
DAnn	JP,C,nn
DBn	IN A,(n)
DCnn	CALL C,nn
DD09	ADD IX,BC
DD19	ADD IX,DE
DD21nn	LD IX,nn
DD22nn	LD (nn),IX
DD23	INC IX
DD29	ADD IX,IX
DD2Ann	LD IX,(nn)
DD2B	DEC IX
DD34d	INC (IX+d)
DD35d	DEC (IX+d)
DD36dn	LD (IX+d),n
DD39	ADD IX,SP
DD46d	LD B,(IX+d)
DD4Ed	LD C,(IX+d)
DD56d	LD D,(IX+d)
DD5Ed	LD E,(IX+d)
DD66d	LD H,(IX+d)
DD6Ed	LD L,(IX+d)
DD70d	LD (IX+d),B
DD71d	LD (IX+d),C
DD72d	LD (IX+d),D
DD73d	LD (IX+d),E
DD74d	LD (IX+d),H
DD75d	LD (IX+d),L
DD77d	LD (IX+d),A
DD7Ed	LD A,(IX+d)
DD86d	ADD A,(IX+d)
DD8Ed	ADC A,(IX+d)
DD96d	SUB (IX+d)
DD9Ed	SBC A,(IX+d)
DDA6d	AND (IX+d)
DDAEd	XOR (IX+d)
DDB6d	OR (IX+d)
DDBEd	CP (IX+d)
DDCBd06	RLC (IX+d)
DDCBd0E	RRC (IX+d)
DDCBd16	RL (IX+d)
DDCBd1E	RR (IX+d)
DDCBd26	SLA (IX+d)
DDCBd2E	SRA (IX+d)
DDCBd3E	SRL (IX+d)
DDCBd46	BIT 0,(IX+d)
DDCBd4E	BIT 1,(IX+d)
DDCBd56	BIT 2,(IX+d)
DDCBd5E	BIT 3,(IX+d)
DDCBd66	BIT 4,(IX+d)
DDCBd6E	BIT 5,(IX+d)
DDCBd76	BIT 6,(IX+d)
DDCBd7E	BIT 7,(IX+d)
DDCBd86	RES 0,(IX+d)
DDCBd8E	RES 1,(IX+d)
DDCBd96	RES 2,(IX+d)
DDCBd9E	RES 3,(IX+d)
DDCBdA6	RES 4,(IX+d)
DDCBdAE	RES 5,(IX+d)
DDCBdB6	RES 6,(IX+d)
DDCBdBE	RES 7,(IX+d)
DDCBdC6	SET 0,(IX+d)
DDCBdCE	SET 1,(IX+d)
DDCBdD6	SET 2,(IX+d)
DDCBdDE	SET 3,(IX+d)
DDCBdE6	SET 4,(IX+d)
DDCBdEE	SET 5,(IX+d)
DDCBdF6	SET 6,(IX+d)
DDCBdFE	SET 7,(IX+d)
DDE1	POP IX
DDE3	EX (SP),IX
DDE5	PUSH IX
DDE9	JP (IX)

(nn) = Address of memory location
nn = Data (16 bit)
n = Data (8-bit)
d = displacement
d2 = d − 2

12.16 Instruction Set: Numerical Order (Continued)

Op Code	Mnemonic
DDF9	LD SP,IX
DEn	SCB A,n
DF	RST 18H
E0	RET PO
E1	POP HL
E2nn	JP PO,nn
E3	EX (SP),HL
E4nn	CALL PO,nn
E5	PUSH HL
E6n	AND n
E7	RST 20H
E8	RET PE
E9	JP (HL)
EAnn	JP PE,nn
EB	EX DE,HL
ECnn	CALL PE,nn
ED40	IN B,(C)
ED41	OUT (C),B
ED42	SBC HL,BC
ED43nn	LD (nn),BC
ED44	NEG
ED45	RETN
ED46	IM 0
ED47	LD I,A
ED48	IN C,(C)
ED49	OUT (C),C
ED4A	ADC HL,BC
ED4Bnn	LD BC,(nn)
ED4D	RETI
ED50	IN D,(C)
ED51	OUT (C),D
ED52	SBC HL,DE
ED53nn	LD (nn),DE
ED56	IM 1
ED57	LD A,I
ED58	IN E,(C)
ED59	OUT (C), E
ED5A	ADC HL,DE
ED5Bnn	LD DE,(nn)
ED5E	IM 2
ED60	IN H,(C)
ED61	OUT (C),H
ED62	SBC HL,HL
ED67	RRD
ED68	IN L,(C)
ED69	OUT (C),L
ED6A	ADC HL,HL
ED6F	RLD
ED72	SBC HL,SP
ED73nn	LD (nn),SP
ED78	IN A,(C)
ED79	OUT (C),A
ED7A	ADC HL,SP

Op Code	Mnemonic
ED7Bnn	LD SP,(nn)
EDA0	LDI
EDA1	CPI
EDA2	INI
EDA3	OUTI
EDA8	LDD
EDA9	CPD
EDAA	IND
EDAB	OUTD
EDB0	LDIR
EDB1	CPIR
EDB2	INIR
EDB3	OTIR
EDB8	LDDR
EDB9	CPDR
EDBA	INDR
EDBB	OTDR
EEn	XOR n
EF	RST 28H
F0	RET P
F1	POP AF
F2nn	JP P,nn
F3	DI
F4nn	CALL P,nn
F5	PUSH AF
F6n	OR n
F7	RST 30H
F8	RET M
F9	LD SP,HL
FAnn	JP M,nn
FB	EI
FCnn	CALL M,nn
FD09	ADD IY,BC
FD19	ADD IY,DE
FD21nn	LD IY,nn
FD22nn	LD (nn),IY
FD23	INC IY
FD29	ADD IY,IY
FD2Ann	LD IY,(nn)
FD2B	DEC IY
FD34d	INC (IY+d)
FD35d	DEC (IY+d)
FD36dn	LD (IY+d),n
FD39	ADD IY,SP
FD46d	LD B,(IY+d)
FD4Ed	LD C,(IY+d)
FD56d	LD D,(IY+d)
FD5Ed	LD E,(IY+d)
FD66d	LD H,(IY+d)
FD6Ed	LD L,(IY+d)
FD70d	LD (IY+d),B
FD71d	LD (IY+d),C
FD72d	LD (IY+d),D

Op Code	Mnemonic
FD73d	LD (IY+d),E
FD74d	LD (IY+d),H
FD75d	LD (IY+d),L
FD77d	LD (IY+d),A
FD7Ed	LD A,(IY+d)
FD86d	ADD A,(IY+d)
FD8Ed	ADC A,(IY+d)
FD96d	SUB (IY+d)
FD9Ed	SBC A,(IY+d)
FDA6d	AND (IY+d)
FDAEd	XOR (IY+d)
FDB6d	OR (IY+d)
FDBEd	CP (IY+d)
FDE1	POP IY
FDE3	EX (SP), IY
FDE5	PUSH IY
FDE9	JP (IY)
FDF9	LD SP,IY
FDCBd06	RLC (IY+d)
FDCBd0E	RRC (IY+d)
FDCBd16	RL (IY+d)
FDCBd1E	RR (IY+d)
FDCBd26	SLA (IY+d)
FDCBd2E	SRA (IY+d)
FDCBd3E	SRL (IY+d)
FDCBd46	BIT 0,(IY+d)
FDCBd4E	BIT 1,(IY+d)
FDCBd56	BIT 2,(IY+d)
FDCBd5E	BIT 3,(IY+d)
FDCBd66	BIT 4,(IY+d)
FDCBd6E	BIT 5,(IY+d)
FDCBd76	BIT 6,(IY+d)
FDCBd7E	BIT 7,(IY+d)
FDCBd86	RES 0,(IY+d)
FDCBd8E	RES 1,(IY+d)
FDCBd96	RES 2,(IY+d)
FDCBd9E	RES 3,(IY+d)
FDCBdA6	RES 4,(IY+d)
FDCBdAE	RES 5,(IY+d)
FDCBdB6	RES 6,(IY+d)
FDCBdBE	RES 7,(IY+d)
FDCBdC6	SET 0,(IY+d)
FDCBdCE	SET 1,(IY+d)
FDCBdD6	SET 2,(IY+d)
FDCBdDE	SET 3,(IY+d)
FDCBdE6	SET 4,(IY+d)
FDCBdEE	SET 5,(IY+d)
FDCBdF6	SET 6,(IY+d)
FDCBdFE	SET 7,(IY+d)
FEn	CP n
FF	RST 38H

(nn) = Address of memory location   d = displacement
nn = Data (16 bit)   d2 = d − 2
n = Data (8-bit)

13.0 Data Acquisition System

A natural application for the NSC800 is one that requires remote operation. Since power consumption is low if the system consists of only CMOS components, the entire package can conceivably operate from only a battery power source. In the application described herein, the only source of power will be from a battery pack composed of a stacked array of NiCad batteries (see *Figure 20*).

The application is that of a remote data acquisition system. Extensive use is made of some of the other LSI CMOS components manufactured by National: notably the ADC0816 and MM58167. The ADC0816 is a 16-channel analog-to-digital converter which operates from a 5V source. The MM58167 is a microprocessor-compatible real-time clock (RTC). The schematic for this system is shown in *Figure 20*. All the necessary features of the system are contained in six integrated circuits: NSC800, NSC810A, NSC831, HN6136P, ADC0816, and MM58167. Some other small scale integration CMOS components are used for normal interface requirements. To reduce component count, linear selection techniques are used to generate chip selects for the NSC810A and NSC831. Included also is a current loop communication link to enable the remote system to transfer data collected to a host system.

In order to keep component count low and maximize effectiveness, many of the features of the NSC800 family have been utilized. The RAM section of the NSC810A is used as a data buffer to store intermediate measurements and as scratch pad memory for calculations. Both timers contained in the NSC810A are used to produce the clocks required by the A/D converter and the RTC. The Power-Save feature of the NSC800 makes it possible to reduce system power consumption when it is not necessary to collect any data. One of the analog input channels of the A/D is connected to the battery pack to enable the CPU to monitor its own voltage supply and notify the host that a battery change is needed.

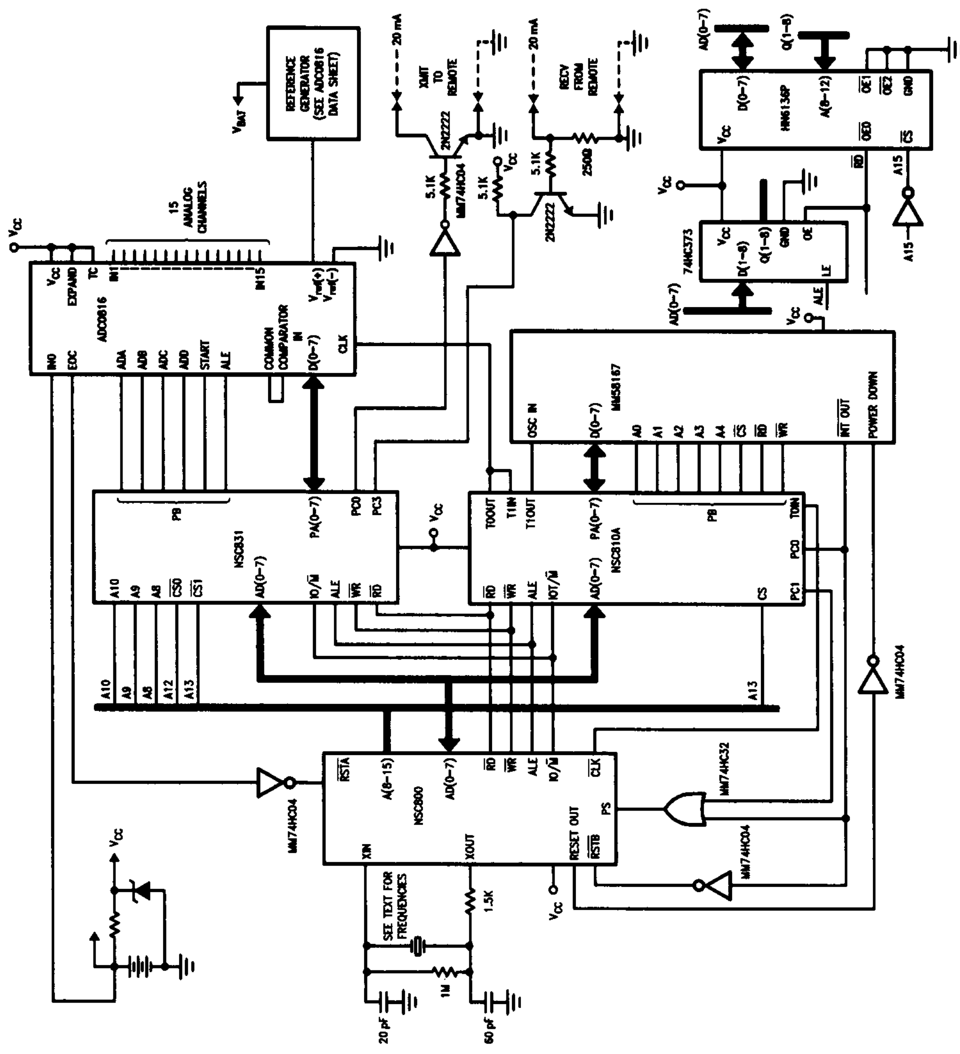
In operation, the NSC800 makes readings on various input conditions through the ADC0816. The type of devices connected to the A/D input depends on the nature of the remote environment. For example, the duties of the remote system might be to monitor temperature variations in a large building. In this case, the analog inputs would be connected to temperature transducers. If the system is situated in a process control environment, it might be monitoring fluid flow, temperatures, fluid levels, etc. In either case, operation would be necessary even if a power failure occurred, thus the need for battery operation or at least battery backup. At some fixed times or at some particular time durations, the system takes readings by selecting one of the analog input channels, commands the A/D to perform a conversion, reads the data, and then formats it for transmission; or, the system checks the readings against set points and transmits a warning if the set points are exceeded. With the addition of the RTC, the host need not command the remote system to take these readings each time it is necessary. The NSC800 could simply set up the RTC to interrupt it at a previously defined time and when the interrupt occurs, make the readings. The resultant values could be stored in the NSC810A for later correlation. In the example of temperature monitoring in a building, it might be desired to know the high and low temperatures for a 12-hour period. After compiling the information, the system could dump the data to the host over the communications link. Note from the schematic that the current for the communication link is supplied by the host to remove the constant current drain from the battery supply.

The required clocks for the two peripheral devices are generated by the two timers in the NSC810A. Through the use of various divisors, the master clock generated by the NSC800 is divided down to produce the clocks. Four examples are shown in the table following *Figure 20*.

All the crystal frequencies are standard frequencies. The various divisors listed are selected to produce, from the master clock frequency of the NSC800, an exact 32,768 Hz clock for the MM58167 and a clock within the operating range of the A/D converter.

The MM58167 is a programmable real-time clock that is microprocessor compatible. Its data format is BCD. It allows the system to program its interrupt register to produce an interrupt output either on a time of day match (which includes the day of the week, the date and month) and/or every month, week, day, hour, minute, second, or tenth of a second. With this capability added to the system, precise time of day measurements are possible without having the CPU do timekeeping. The interrupt output can be connected, through the use of one port bit of the NSC810A, to put the CPU in the power-save mode and reenable it at a preset time. The interrupt output is also connected to one of the hardware restart inputs ($\overline{\text{RSTB}}$) to enable time duration measurements. This power-down mode of operation would not be possible if the NSC800 had the duties of timekeep-

13.0 Data Acquisition System (Continued)



TL/C/5171-34

FIGURE 20. Remote Data Acquisition

13.0 Data Acquisition System (Continued)

ing. When in the power-save mode, the system power requirements are decreased by about 50%, thus extending battery life.

Communication with the peripheral devices (MM58167 and ADC0816) is accomplished through the I/O ports of the NSC810A and NSC831. The peripheral devices are not connected to the bus of the NSC800 as they are not directly compatible with a multiplexed bus structure. Therefore, additional components would be required to place them on the microprocessor bus. Writing data into the MM58167 is performed by first putting the desired data on Port A, followed by selecting the address of the internal register and applying the chip select through the use of Port B. A bit set and clear operation is performed to emulate a pulse on the bit of Port B connected to the $\overline{\text{WR}}$ input of the MM58167. For a read operation, the same sequence of operations is performed except that Port A is set for the input mode of operation and the $\overline{\text{RD}}$ line is pulsed. Similar techniques are used to read converted data from the A/D converter. When a conversion is desired, the CPU selects a channel and commands the ADC0816 to start a conversion. When the conversion is complete, the converter will produce an End-of-Conversion signal which is connected to the $\overline{\text{RSTA}}$ interrupt input of the NSC800.

When operating, the system shown consumes about 125 mw. When in the power-save mode, power consumption is decreased to about 70 mw. If, as is likely, the system is in the power-save mode most of the time, battery life can be quite long depending on the amp-hour rating of the batteries incorporated into the system. For example, if the battery pack is rated at 5 amp-hours, the system should be able to operate for about 400-500 hours before a battery charge or change is required.

As shown in the schematic (refer to *Figure 20*), analog input IN0 is connected to the battery source. In this way, the CPU can monitor its own power source and notify the host that it needs a battery replacement or charge. Since the battery source shown is a stacked array of 7 NiCads producing 8.4V, the converter input is connected in the middle so that it can take a reading on two or three of the cells. Since NiCad batteries have a relatively constant voltage output until very nearly discharged, the CPU can sense that the "knee" of the discharge curve has been reached and notify the host.

Typical Timer Output Frequencies

Crystal Frequency	CPU Clock Output	Timer 0 Output	Timer 1 Output
2.097152 MHz	1.048576 MHz	262.144 kHz divisor = 4	32.768 kHz divisor = 8
3.276800 MHz	1.638400 MHz	327.680 kHz divisor = 5	32.768 kHz divisor = 10
4.194304 MHz	2.097152 MHz	262.144 kHz divisor = 8	32.768 kHz divisor = 8
4.915200 MHz	2.457600 MHz	491.520 kHz divisor = 5	32.768 kHz divisor = 15

14.0 NSC800M/883B MIL-STD-833 Class C Screening

National Semiconductor offers the NSC800D and NSC800E with full class B screening per MIL-STD-883 for Military/Aerospace programs requiring high reliability. In addition, this screening is available for all of the key NSC800 peripheral devices.

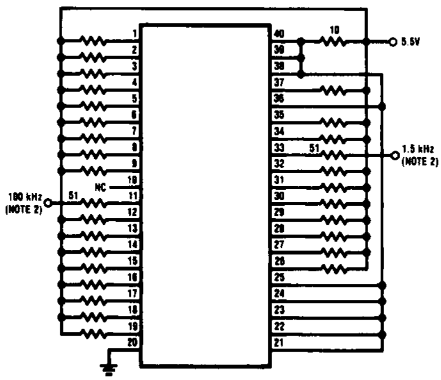
Electrical testing is performed in accordance with RESTS800X, which tests or guarantees all of the electrical performance characteristics of the NSC800 data sheet. A copy of the current revision of RETS800X is available upon request.

100% Screening Flow

Test	MIL-STD-883 Method/Condition	Requirement
Internal Visual	2010B	100%
Stabilization Bake	1008 C 24 Hrs. @ +150°C	100%
Temperature Cycling	1010 C 10 Cycles −65°C/+150°C	100%
Constant Acceleration	2001 E 30,000 G's, Y1 Axis	100%
Fine Leak	1014 A or B	100%
Gross Leak	1014C	100%
Burn-In	1015 160 Hrs. @ +125°C (using burn-in circuits shown below)	100%
Final Electrical	+25°C DC per RETS800X	100%
PDA	10% Max	
	+125°C AC and DC per RETS800X	100%
	−55°C AC and DC per RETS800X	100%
	+25°C AC per RETS800X	100%
QA Acceptance	5005	Sample Per
Quality Conformance		Method 5005
External Visual	2009	100%

15.0 Burn-In Circuits

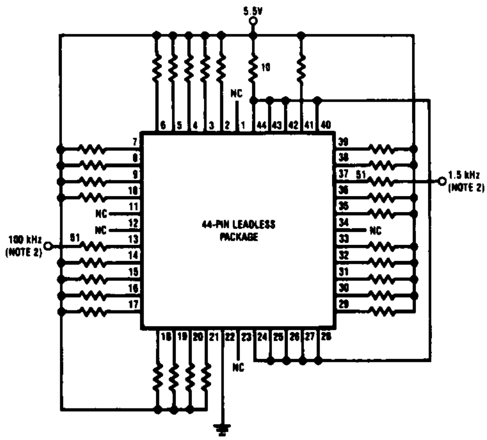
5240HR
NSC800D/883B (Dual-In-Line)



TL/C/5171-32

Top View

5241HR
NSC800E/883B (Leadless Chip Carrier)



TL/C/5171-33

All resistors 2.7 kΩ unless marked otherwise.

Note 1: All resistors are ¼W ± 5% unless otherwise specified.

Note 2: All clocks 0V to 3V, 50% duty cycle, in phase with < 1 μs rise and fall time.

Note 3: Device to be cooled down under power after burn-in.

16.0 Ordering Information

NSC800 X X X X

(first X — package)
- **D** = Ceramic Package
- **N** = Plastic Package
- **E** = Ceramic Leadless Chip Carrier (LCC)
- **V** = Plastic Leaded Chip Carrier (PCC)

(second X — clock)
- **−4** = 4 MHz Clock
- **−35** = 3.5 MHz Clock Output
- **−3** = 2.5 MHz Clock Output
- **−1** = 1 MHz Clock Output

(third X — temperature)
- **I** = Industrial Temperature (−40°C to +85°C)
- **M** = Military Temperature (−55°C to +125°C)
- **MIL** = Special Temperature (−55°C to +90°C)
- **No Designation** = Commercial Temperature (0°C to +70°C)

(fourth X — screening)
- **/A+** = A+ Reliability Screening
- **/883** = MIL-STD-883 Screening (Note 1)

Note 1: Do not specify a temperature option; all parts are screened to military temperature.

17.0 Reliability Information

Gate Count 2750

Transistor Count 11,000