ST6 FAMILY PROGRAMMING MANUAL

Rev. 2.0

October 2004

ST 62xx	**ST6200/01/03/08/09/10/15/20/25 ST6218/28 ST6252/53/55/60/62/63/65**

Ceibo Emulator Supporting ST62xx:	**EB-ST62** **http://ceibo.com/eng/products/ebst62.shtml**

PROGRAMMING MANUAL

ST6 FAMILY

INTRODUCTION

This manual deals with the description of the instruction set and addressing modes of ST6 microcontroller series. The manual is divided in two main sections. The first one includes, after a general family description, the addressing modes description. The second section includes the detailed description of ST6 instruction set. Each instruction is described in detail with the differences between each ST6 series.

Table 1. ST6 Series Core Characteristics

	ST6 Series
Stack Levels	6
Interrupt Vectors	5
NMI	YES
Flags Sets	3
Program ROM	2K + 2K∗n 20K Max
Data RAM	64 byte*m
Data ROM	64 byte pages in ROM
Carry Flag SUB Instruction	Reset if A > Source
Carry Flag CP Instruction	Set if A < Source

Rev. 2.0

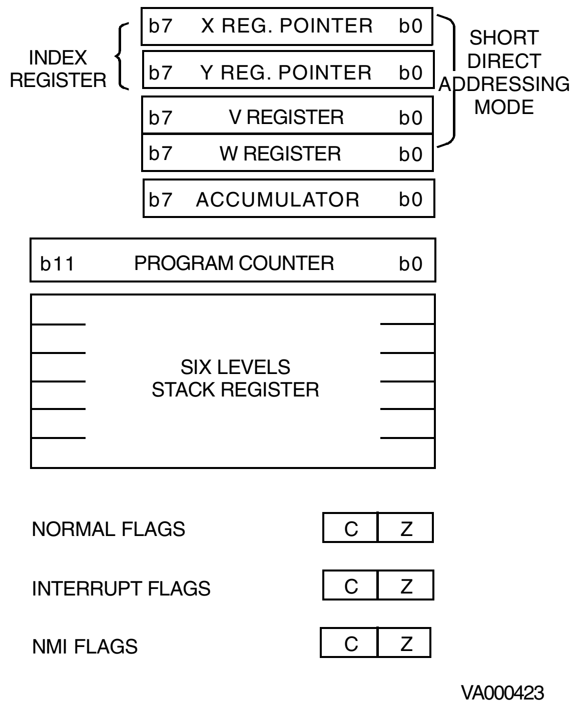
Table of Contents

1 PROGRAMMING MODEL

It is useful at this stage to outline the programming model of the ST6 series, by which we mean the available memory spaces, their relation to one another, the interrupt philosophy and so on.

Memory Spaces. The ST6 devices have three different memory spaces: data, program and stack. All addressing modes are memory space specific so there is no need for the user to specify which space is being used as in more complex systems. The stack space, which is used automatically with subroutine and interrupt management for program counter storage, is not accessible to the user.

Figure 1. ST6 Family Programming Model



PROGRAMMING MODEL (Cont'd)

Figure 2. ST6 Data Space Example

b7 … b0	Address
NOT IMPLEMENTED	000h 03Fh
DATA ROM/EPROM WINDOW 64 BYTE	040h 07Fh
X REGISTER	080h
Y REGISTER	081h
V REGISTER	082h
W REGISTER	083h
DATA RAM 60 BYTES	084h 0BFh
PORT A DATA REGISTER - DRA	0C0h
PORT B DATA REGISTER - DRB	0C1h
PORT C DATA REGISTER - DRC	0C2h
RESERVED	0C3h
PORT A DATA DIRECTION REGISTER - DDRA	0C4h
PORT B DATA DIRECTION REGISTER - DDRB	0C5h
PORT C DATA DIRECTION REGISTER - DDRC	0C6h
RESERVED	0C7h
INTERRUPT OPTION REGISTER - IOR	0C8h
DATA ROM WINDOW REGISTER - DRWR	0C9h
RESERVED	0CAh 0CBh
PORT A OPTION REGISTER - ORA	0CCh
PORT B OPTION REGISTER -ORB	0CDh
PORT C OPTION REGISTER - ORC	0CEh
RESERVED	0CFh
A/D DATA REGISTER - ADR	0D0h
A/D CONTROL REGISTER - ADCR	0D1h
TIMER PSC REGISTER - PSCR	0D2h
TIMER COUNTER REGISTER - TCR	0D3h
TIMER STATUS CONTROL REGISTER - TSCR	0D4h
RESERVED	0D5h 0D6h 0D7h
WATCHDOG REGISTER - WDGR	0D8h
RESERVED	0D9h 0FEh
ACCUMULATOR	OFFh

Figure 3. ST6 Program Memory Example

b7 … b0	Address
NOT IMPLEMENTED	0000h 07FFh
RESERVED	0800h 087Fh
USER PROGRAM ROM 1828 BYTES	0880h 0F9Fh
RESERVED	0FA0h 0FEFh
INTERRUPT VECTOR #4 A/D INTERRUPT	0FF0h 0FF1h
INTERRUPT VECTOR #3 TIMER INTERRUPT	0FF2h 0FF3h
INTERRUPT VECTOR #2 PORT B & C INTERRUPT	0FF4h 0FF5h
INTERRUPT VECTOR #1 PORT A INTERRUPT	0FF6h 0FF7h
RESERVED	0FF8h 0FFBh
INTERRUPT VECTOR #0 NMI INTERRUPT	0FFCh 0FFDh
USER RESET VECTOR	0FFEh 0FFFh

On EPROM versions there are no re-served areas. These reserved bytes are present on ROM/OTP versions.

Data Memory Space. The following registers in the data space have fixed addresses which are hardware selected so as to decrease access times and reduce addressing requirements and hence program length. The Accumulator is an 8-bit register in location 0FFh. The X, Y, V & W registers have the addresses 80h-83h respectively. These are used for short direct addressing, reducing byte requirements in the program while the first two, X & Y, can also be used as index registers in the indirect addressing mode. These registers are part of the data RAM space. In the ST6 for data space ROM a 6-bit (64 bytes addressing) window multiplexing in program ROM is available through a dedicated data ROM banking register.

PROGRAMMING MODEL (Cont'd)

For data RAM and I/O expansion the lowest 64 bytes of data space (00h-03Fh) are paged through a data RAM banking register.

Self-check Interrupt Vector FF8h & FF9h:jp (self-check interrupt routine)

A jump instruction to the reset and interrupt routines must be written into these locations.

ST6 Program Memory Space. The ST6 devices can directly address up to 4K bytes (program counter is 12 bits wide). A greater ROM size is obtained by paging the lower 2K of the program ROM through a dedicated banking register located in the data space. The higher 2K of the program ROM can be seen as static and contains the reset, NMI and interrupt vectors at the following fixed locations:

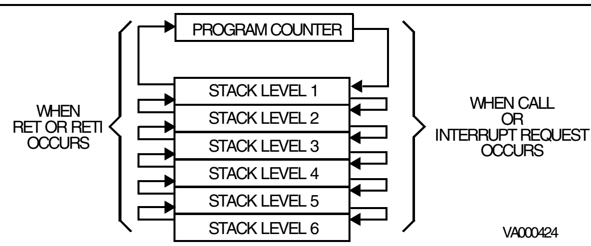
Reset Vector	FFEh & FFFh:jp (reset routine)
NMI Interrupt Vector	FFCh & FFDh:jp (NMI routine)
Non user Vector	FFAh & FFBh
Non user Vector	FF8h & FF9h
Interrupt #1 Vector	FF6h & FF7h jp (Int 1 routine)
Interrupt #2 Vector	FF4h & FF5h jp (Int 2 routine)
Interrupt #3 Vector	FF2h & FF3h jp (Int 3 routine)
Interrupt #4 Vector	FF0h & FF1h jp (Int 4 routine)

Program Counter & Stack Area. The program counter is a 12-bit counter register since it has to cover a direct addressing of 4K byte program memory space. When an interrupt or a subroutine occurs the current PC value is forward "pushed" into a deep LIFO stacking area. On the return from the routine the top (last in) PC value is "popped" out and becomes the current PC value. The ST60/61 series offer a 4-word deep stack for program counter storage during interrupt and sub-routines calls. In the ST6 series the stack is 6-word deep.

Status Flags. Three pairs of status flags, each pair consisting of a Zero flag and a Carry flag, are available. In the ST6 an additional third set is available. One pair monitors the normal status while the second monitors the state during interrupts; the third flags set monitors the status during Non Maskable interrupt servicing. The switching from one set to another is automatic as the interrupt requests (or NMI request for ST6 only) are acknowledged and when the program returns after an interrupt service routine. After reset, the NMI set is active, until the first RETI instruction is executed.

ST6 Interrupt Description. The ST6 devices have 5 user interrupt vectors (plus one vector for testing purposes). Interrupt vector #0 is connected to the not maskable interrupt input of the core. Interrupts from #1 to #4 can be connected to different on-chip and external sources (see individual datasheets for detailed information). All interrupts can be globally disabled through the interrupt option register. After the reset ST6 devices are in NMI mode, so no other interrupts can be accepted and the NMI flags set is in use, until the RETI instruction is performed. If an interrupt is detected, a special cycle is executed. During this cycle, the program counter is loaded with the related interrupt vector address. NMI can interrupt other interrupt routines at any time while normal interrupt can't interrupt each other. If more than one interrupt is awaiting service, they will be accepted according to their priority. Interrupt #1 has the highest priority while interrupt #4 the lowest. This priority relationship is fixed.

Figure 4. ST6 Stack Area



2 ADDRESSING MODES

The ST6 core offers nine addressing modes, which are described in the following paragraphs. Three different address spaces are available: Program space, Data space, and Stack space. Program space contains the instructions which are to be executed, plus the data for immediate mode instructions. Data space contains the Accumulator, the X,Y,V and W registers, peripheral and Input/Output registers, the RAM locations and Data ROM locations (for storage of tables and constants). Stack space contains six 12-bit RAM cells used to stack the return addresses for subroutines and interrupts.

Immediate. In the immediate addressing mode, the operand of the instruction follows the opcode location. As the operand is a ROM byte, the immediate addressing mode is used to access constants which do not change during program execution (e.g., a constant used to initialize a loop counter).

Direct. In the direct addressing mode, the address of the byte which is processed by the instruction is stored in the location which follows the opcode. Direct addressing allows the user to directly address the 256 bytes in Data Space memory with a single two-byte instruction.

Short Direct. The core can address the four RAM registers X,Y,V,W (locations 80h, 81h, 82h, 83h) in the short-direct addressing mode. In this case, the instruction is only one byte and the selection of the location to be processed is contained in the opcode. Short direct addressing is a subset of the direct addressing mode. (Note that 80h and 81h are also indirect registers).

Extended. In the extended addressing mode, the 12-bit address needed to define the instruction is obtained by concatenating the four less significant bits of the opcode with the byte following the opcode. The instructions (JP, CALL) which use the extended addressing mode are able to branch to any address of the 4K bytes Program space.

An extended addressing mode instruction is 2-bytes long.

Program Counter Relative. The relative addressing mode is only used in conditional branch instructions. The instruction is used to perform a test and, if the condition is true, a branch with a span of -15 to +16 locations around the address of the relative instruction. If the condition is not true, the instruction which follows the relative instruction is executed. The relative addressing mode instruction is one-byte long. The opcode is obtained by adding the three most significant bits which characterize the kind of the test, one bit which determines whether the branch is a forward (when it is 0) or backward (when it is 1) branch and the four less significant bits which give the span of the branch (0h to Fh) which must be added or subtracted to the address of the relative instruction to obtain the address of the branch.

Bit Direct. In the bit direct addressing mode, the bit to be set or cleared is part of the opcode, and the byte following the opcode points to the address of the byte in which the specified bit must be set or cleared. Thus, any bit in the 256 locations of Data space memory can be set or cleared.

Bit Test & Branch. The bit test and branch addressing mode is a combination of direct addressing and relative addressing. The bit test and branch instruction is three-byte long. The bit identification and the tested condition are included in the opcode byte. The address of the byte to be tested follows immediately the opcode in the Program space. The third byte is the jump displacement, which is in the range of -127 to +128. This displacement can be determined using a label, which is converted by the assembler.

Indirect. In the indirect addressing mode, the byte processed by the register-indirect instruction is at the address pointed by the content of one of the indirect registers, X or Y (80h,81h). The indirect register is selected by the bit 4 of the opcode. A register indirect instruction is one byte long.

Inherent. In the inherent addressing mode, all the information necessary to execute the instruction is contained in the opcode. These instructions are one byte long.

3 ST6 INSTRUCTION SET

The ST6 core offers a set of 40 basic instructions which, when combined with nine addressing modes, yield 244 usable opcodes. They can be divided into six different types: load/store, arithmetic/logic, conditional branch, control instructions, jump/call, and bit manipulation. The following paragraphs describe the different types.

All the instructions belonging to a given type are presented in individual tables.

Load & Store. These instructions use one, two or three bytes in relation with the addressing mode. One operand is the Accumulator for LOAD and the other operand is obtained from data memory using one of the addressing modes.

For Load Immediate one operand can be any of the 256 data space bytes while the other is always immediate data.

Table 2. Load & Store Instructions

Instruction	Addressing Mode	Bytes	Cycles	Flags	
				Z	C
LD A, X	Short Direct	1	4	Δ	*
LD A, Y	Short Direct	1	4	Δ	*
LD A, V	Short Direct	1	4	Δ	*
LD A, W	Short Direct	1	4	Δ	*
LD X, A	Short Direct	1	4	Δ	*
LD Y, A	Short Direct	1	4	Δ	*
LD V, A	Short Direct	1	4	Δ	*
LD W, A	Short Direct	1	4	Δ	*
LD A, rr	Direct	2	4	Δ	*
LD rr, A	Direct	2	4	Δ	*
LD A, (X)	Indirect	1	4	Δ	*
LD A, (Y)	Indirect	1	4	Δ	*
LD (X), A	Indirect	1	4	Δ	*
LD (Y), A	Indirect	1	4	Δ	*
LDI A, #N	Immediate	2	4	Δ	*
LDI rr, #N	Immediate	3	4	*	*

Notes:
X,Y. Indirect Register Pointers, V & W Short Direct Registers
. Immediate data (stored in ROM memory)
rr. Data space register
Δ. Affected
* . Not Affected

ST6 INSTRUCTION SET (Cont'd)

Arithmetic and Logic. These instructions are used to perform the arithmetic calculations and logic operations. In AND, ADD, CP, SUB instructions one operand is always the accumulator while the other can be either a data space memory content or an immediate value in relation with the addressing mode. In CLR, DEC, INC instructions the operand can be any of the 256 data space addresses. In COM, RLC, SLA the operand is always the accumulator.

Table 3. Arithmetic & Logic Instructions

Instruction	Addressing Mode	Bytes	Cycles	Flags	
				Z	C
ADD A, (X)	Indirect	1	4	Δ	Δ
ADD A, (Y)	Indirect	1	4	Δ	Δ
ADD A, rr	Direct	2	4	Δ	Δ
ADDI A, #N	Immediate	2	4	Δ	Δ
AND A, (X)	Indirect	1	4	Δ	Δ
AND A, (Y)	Indirect	1	4	Δ	Δ
AND A, rr	Direct	2	4	Δ	Δ
ANDI A, #N	Immediate	2	4	Δ	Δ
CLR A	Short Direct	2	4	Δ	Δ
CLR r	Direct	3	4	*	*
COM A	Inherent	1	4	Δ	Δ
CP A, (X)	Indirect	1	4	Δ	Δ
CP A, (Y)	Indirect	1	4	Δ	Δ
CP A, rr	Direct	2	4	Δ	Δ
CPI A, #N	Immediate	2	4	Δ	Δ
DEC X	Short Direct	1	4	Δ	*
DEC Y	Short Direct	1	4	Δ	*
DEC V	Short Direct	1	4	Δ	*
DEC W	Short Direct	1	4	Δ	*
DEC A	Direct	2	4	Δ	*
DEC rr	Direct	2	4	Δ	*
DEC (X)	Indirect	1	4	Δ	*
DEC (Y)	Indirect	1	4	Δ	*
INC X	Short Direct	1	4	Δ	*
INC Y	Short Direct	1	4	Δ	*
INC V	Short Direct	1	4	Δ	*
INC W	Short Direct	1	4	Δ	*
INC A	Direct	2	4	Δ	*
INC rr	Direct	2	4	Δ	*
INC (X)	Indirect	1	4	Δ	*
INC (Y)	Indirect	1	4	Δ	*
RLC A	Inherent	1	4	Δ	Δ
SLA A	Inherent	2	4	Δ	Δ
SUB A, (X)	Indirect	1	4	Δ	Δ
SUB A, (Y)	Indirect	1	4	Δ	Δ
SUB A, rr	Direct	2	4	Δ	Δ
SUBI A, #N	Immediate	2	4	Δ	Δ

Notes:
X,Y. Indirect Register Pointers, V & W Short Direct Registers
. Immediate data (stored in ROM memory)
rr. Data space register

Δ. Affected
* . Not Affected

ST6 INSTRUCTION SET (Cont'd)

Conditional Branch. The branch instructions achieve a branch in the program when the selected condition is met.

Bit Manipulation Instructions. These instructions can handle any bit in data space memory. One group either sets or clears. The other group (see Conditional Branch) performs the bit test branch operations.

Control Instructions. The control instructions control the MCU operations during program execution.

Jump and Call. These two instructions are used to perform long (12-bit) jumps or subroutine calls inside the whole program space.

Table 4. Conditional Branch Instructions

Instruction	Branch If	Bytes	Cycles	Flags	
				Z	C
JRC e	C = 1	1	2	*	*
JRNC e	C = 0	1	2	*	*
JRZ e	Z = 1	1	2	*	*
JRNZ e	Z = 0	1	2	*	*
JRR b, rr, ee	Bit = 0	3	5	*	Δ
JRS b, rr, ee	Bit = 1	3	5	*	Δ

Notes:
b. 3-bit address
e. 5-bit signed displacement in the range -15 to +16<F128M>
ee. 8-bit signed displacement in the range -126 to +129
rr. Data space register
Δ . Affected. The tested bit is shifted into carry.
* . Not Affected

Table 5. Bit Manipulation Instructions

Instruction	Addressing Mode	Bytes	Cycles	Flags	
				Z	C
SET b,rr	Bit Direct	2	4	*	*
RES b,rr	Bit Direct	2	4	*	*

Notes:
b. 3-bit address
rr. Data space register
* . Not Affected

Table 6. Control Instructions

Instruction	Addressing Mode	Bytes	Cycles	Flags	
				Z	C
NOP	Inherent	1	2	*	*
RET	Inherent	1	2	*	*
RETI	Inherent	1	2	Δ	Δ
STOP (1)	Inherent	1	2	*	*
WAIT	Inherent	1	2	*	*

Notes:
1. This instruction is deactivated and a WAIT is automatically executed instead of a STOP if the watchdog function is selected.
Δ . Affected
*. Not Affected

Table 7. Jump & Call Instructions

Instruction	Addressing Mode	Bytes	Cycles	Flags	
				Z	C
CALL abc	Extended	2	4	*	*
JP abc	Extended	2	4	*	*

Notes:
abc. 12-bit address
* . Not Affected

Opcode Map Summary. The following table contains an opcode map for the instructions used by the ST6

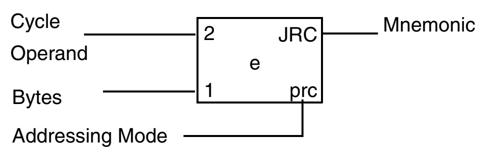
LOW HI	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	LOW HI
0 0000	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b0,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 LD a,(x) 1 ind	**0 0000**
1 0001	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b0,rr,ee 3 bt	2 JRZ e 1 pcr	4 INC x 1 sd	2 JRC e 1 prc	4 LDI a,nn 2 imm	**1 0001**
2 0010	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b4,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 CP a,(x) 1 ind	**2 0010**
3 0011	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b4,rr,ee 3 bt	2 JRZ e 1 pcr	4 LD a,x 1 sd	2 JRC e 1 prc	4 CPI a,nn 2 imm	**3 0011**
4 0100	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b2,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 ADD a,(x) 1 ind	**4 0100**
5 0101	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b2,rr,ee 3 bt	2 JRZ e 1 pcr	4 INC y 1 sd	2 JRC e 1 prc	4 ADDI a,nn 2 imm	**5 0101**
6 0110	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b6,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 INC (x) 1 ind	**6 0110**
7 0111	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b6,rr,ee 3 bt	2 JRZ e 1 pcr	4 LD a,y 1 sd	2 JRC e 1 prc	#	**7 0111**
8 1000	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b1,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 LD (x),a 1 ind	**8 1000**
9 1001	2 RNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b1,rr,ee 3 bt	2 JRZ e 1 pcr	4 INC v 1 sd	2 JRC e 1 prc	#	**9 1001**
A 1010	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b5,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 AND a,(x) 1 ind	**A 1010**
B 1011	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b5,rr,ee 3 bt	2 JRZ e 1 pcr	4 LD a,v 1 sd	2 JRC e 1 prc	4 ANDI a,nn 2 imm	**B 1011**
C 1100	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b3,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 SUB a,(x) 1 ind	**C 1100**
D 1101	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b3,rr,ee 3 bt	2 JRZ e 1 pcr	4 INC w 1 sd	2 JRC e 1 prc	4 SUBI a,nn 2 imm	**D 1101**
E 1110	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b7,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 DEC (x) 1 ind	**E 1110**
F 1111	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b7,rr,ee 3 bt	2 JRZ e 1 pcr	4 LD a,w 1 sd	2 JRC e 1 prc	#	**F 1111**

Abbreviations for Addressing Modes:

dir	Direct
sd	Short Direct
imm	Immediate
inh	Inherent
ext	Extended
b.d	Bit Direct
bt	Bit Test
pcr	Program Counter Relative
ind	Indirect

Legend:

#	Indicates Illegal Instructions
e	5-Bit Displacement
b	3-Bit Address
rr	1byte dataspace address
nn	1 byte immediate data
abc	12-bit address
ee	8-bit Displacement



Opcode Map Summary (Continued)

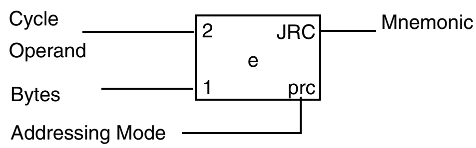
LOW HI	**8** **1000**	**9** **1001**	**A** **1010**	**B** **1011**	**C** **1100**	**D** **1101**	**E** **1110**	**F** **1111**	LOW HI
0 **0000**	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b0,rr 2 b.d	2 JRZ e 1 pcr	4 LDI rr,nn 3 imm	2 JRC e 1 prc	4 LD a,(y) 1 ind	**0** **0000**
1 **0001**	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b0,rr 2 b.d	2 JRZ e 1 pcr	4 DEC x 1 sd	2 JRC e 1 prc	4 LD a,rr 2 dir	**1** **0001**
2 **0010**	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b4,rr 2 b.d	2 JRZ e 1 pcr	4 COM a	2 JRC e 1 prc	4 CP a,(y) 1 ind	**2** **0010**
3 **0011**	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b4,rr 2 b.d	2 JRZ e 1 pcr	4 LD x,a 1 sd	2 JRC e 1 prc	4 CP a,rr 2 dir	**3** **0011**
4 **0100**	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b2,rr 2 b.d	2 JRZ e 1 pcr	2 RETI 1 inh	2 JRC e 1 prc	4 ADD a,(y) 1 ind	**4** **0100**
5 **0101**	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b2,rr 2 b.d	2 JRZ e 1 pcr	4 DEC y 1 sd	2 JRC e 1 prc	4 ADD a,rr 2 dir	**5** **0101**
6 **0110**	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b6,rr 2 b.d	2 JRZ e 1 pcr	2 STOP 1 inh	2 JRC e 1 prc	4 INC (y) 1 ind	**6** **0110**
7 **0111**	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b6,rr 2 b.d	2 JRZ e 1 pcr	4 LD y,a 1 sd	2 JRC e 1 prc	4 INC rr 2 dir	**7** **0111**
8 **1000**	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b1,rr 2 b.d	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 LD (y),a 1 ind	**8** **1000**
9 **1001**	2 RNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b1,rr 2 b.d	2 JRZ e 1 pcr	4 DEC v 1 sd	2 JRC e 1 prc	4 LD rr,a 2 dir	**9** **1001**
A **1010**	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b5,rr 2 b.d	2 JRZ e 1 pcr	4 RCL a 1 inh	2 JRC e 1 prc	4 AND a,(y) 1 ind	**A** **1010**
B **1011**	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b5,rr 2 b.d	2 JRZ e 1 pcr	4 LD v,a 1 sd	2 JRC e 1 prc	4 AND a,rr 2 dir	**B** **1011**
C **1100**	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b3,rr 2 b.d	2 JRZ e 1 pcr	2 RET 1 inh	2 JRC e 1 prc	4 SUB a,(y) 1 ind	**C** **1100**
D **1101**	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b3,rr 2 b.d	2 JRZ e 1 pcr	4 DEC w 1 sd	2 JRC e 1 prc	4 SUB a,rr 2 dir	**D** **1101**
E **1110**	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b7,rr 2 b.d	2 JRZ e 1 pcr	2 WAIT 1 inh	2 JRC e 1 prc	4 DEC (y) 1 ind	**E** **1110**
F **1111**	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b7,rr 2 b.d	2 JRZ e 1 pcr	4 LD w,a 1 sd	2 JRC e 1 prc	4 DEC rr 2 dir	**F** **1111**

Abbreviations for Addressing Modes:

dir	Direct
sd	Short Direct
imm	Immediate
inh	Inherent
ext	Extended
b.d	Bit Direct
bt	Bit Test
pcr	Program Counter Relative
ind	Indirect

Legend:

#	Indicates Illegal Instructions
e	5-Bit Displacement
b	3-Bit Address
rr	1byte dataspace address
nn	1 byte immediate data
abc	12-bit address
ee	8-bit Displacement



ADD   Addition   ADD

Mnemonic: ADD

Function: Addition

Description: The contents of the source byte is added to the accumulator leaving the result in the accumulator. The source register remains unaltered.

Operation: dst ← dst + src
The destination must be the accumulator.

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
ADD dst,src				Z	C
ADD A,A	5F FF	2	4	Δ	Δ
ADD A,X	5F 80	2	4	Δ	Δ
ADD A,Y	5F 81	2	4	Δ	Δ
ADD A,V	5F 82	2	4	Δ	Δ
ADD A,W	5F 83	2	4	Δ	Δ
ADD A,(X)	47	1	4	Δ	Δ
ADD A,(Y)	4F	1	4	Δ	Δ
ADD A,rr	5F rr	2	4	Δ	Δ

Notes:

rr. 1 Byte dataspace address.

Δ: Z is set if the result is zero. Cleared otherwise.
C is cleared before the operation and than set if there is an overflow from the 8-bit result.

Example: If data space register 22h contains the value 33h and the accumulator holds the value 20h then the instruction,

ADD A,22h

will cause the accumulator to hold 53h (i.e. 33+20).

Addressing Modes: Source: Direct, Indirect

Destination: Accumulator

ADDI — Addition Immediate — ADDI

Mnemonic: ADDI

Function: Addition Immediate

Description: The immediately addressed data (source) is added to the accumulator leaving the result in the accumulator.

Operation: dst 🡐 dst + src

The destination must be the accumulator.

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
ADDI dst,src				**Z**	**C**
ADDI A,nn	57 nn	2	4	Δ	Δ

Notes:

nn. 1 Byte immediate data

Δ: Z is set if result is zero. Cleared otherwise
C is cleared before the operation and than set if there is an overflow from the 8-bit result

Example: If the accumulator holds the value 20h then the instruction,

ADDI A,22h

will cause the accumulator to hold 42h (i.e. 22+20).

Addressing Modes: Source: Immediate

Destination: Accumulator

AND — Logical AND — AND

Mnemonic: AND

Function: Logical AND

Description: This instruction logically ANDs the source register and the accumulator. The result is left in the destination register and the source is unaltered.

Operation: dst ← src AND dst

The destination must be the accumulator.

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
AND dst,src				Z	C
AND A,A	BF FF	2	4	Δ	*
AND A,X	BF 80	2	4	Δ	*
AND A,Y	BF 81	2	4	Δ	*
AND A,V	BF 82	2	4	Δ	*
AND A,W	BF 83	2	4	Δ	*
AND A,(X)	A7	1	4	Δ	*
AND A,(Y)	AF	1	4	Δ	*
AND A,rr	BF rr	2	4	Δ	*

Notes:

rr. 1 Byte dataspace address

*. C is unaffected

Δ. Z is set if the result is zero. Cleared otherwise.

Example: If data space register 54h contains the binary value 11110000 and the accumulator contains the binary value 11001100 then the instruction,

AND A,54h

will cause the accumulator to be altered to 11000000.

Addressing Modes: Source: Direct, Indirect.

Destination: Accumulator

ANDI — Logical AND Immediate — ANDI

Mnemonic: ANDI

Function: Logical AND Immediate

Description: This instruction logically ANDs the immediate data byte and the accumulator. The result is left in the accumulator.

Operation: dst ← src AND dst

The source is immediate data and the destination must be the accumulator.

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
ANDI dst,src				Z	C
ANDI A,nn	B7 nn	2	4	Δ	*

Notes:

nn. 1 Byte immediate data

*. C is unaffected

Δ. Z is set if the result is zero. Cleared otherwise.

Example: If the accumulator contains the binary value 00001111 then the instruction,

ANDI A,33h

will cause the accumulator to hold the value 00000011.

Addressing Modes: Source: Immediate

Destination: Accumulator

CALL — Call Subroutine — CALL

Mnemonic: CALL

Function: Call Subroutine

Description: The CALL instruction is used to call a subroutine. It "pushes" the current contents of the program counter (PC) onto the top of the stack. The specified destination address is then loaded into the PC and points to the first instruction of a procedure. At the end of the procedure a RETurn instruction can be used to return to the original program flow. RET pops the top of the stack back into the PC. Because the ST6 stack is 4 levels deep (ST60) and 6 levels deep (ST62,ST63), a maximum of four/six calls or interrupts may be nested. If more calls are nested, the latest stacked PC values will be lost. In this case returns will return to the PC values stacked first.

Operation: PC ← dst; Top of stack ← PC

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
CALL dst				**Z**	**C**
CALL abc	c0001 ab	2	4	*	*

Notes:

abc. the three half bytes of a 12-bit address, the start location of the subroutine.

*. C,Z not affected

Example: If the current PC is 345h then the instruction,

CALL 8DCh

The current PC 345h is pushed onto the top of the stack and the PC will be loaded with the value 8DCh. The next instruction to be executed will be the instruction at 8DCh, the first instruction of the called subroutine.

Addressing Modes: Extended

CLR Clear CLR

Mnemonic: CLR

Function: Clear

Description: The destination register is cleared to 00h.

Operation: dst ← 0

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
CLR dst				Z	C
CLR A	DF FF	2	4	Δ	Δ
CLR X	0D 80 00	3	4	*	*
CLR Y	0D 81 00	3	4	*	*
CLR V	0D 82 00	3	4	*	*
CLR W	0D 83 00	3	4	*	*
CLR rr	0D rr 00	3	4	*	*

Notes:

rr. 1 Byte dataspace address

Δ. Z set, Δ. C reset

*. C,Z unaffected

Example: If data space register 22h contains the value 33h,

CLR 22h

will cause register 22h to hold 00h.

Addressing Modes: Direct

COM **Complement** COM

Mnemonic: COM

Function: Complement

Description: This instruction complements each bit of the accumulator; all bits which are set to 1 are cleared to 0 and vice-versa.

Operation: dst ← NOT dst

The destination must be the accumulator.

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
COM dst				Z	C
COM A	2D	1	4	Δ	Δ

Note:

Δ: Z is set if the result is zero. Cleared otherwise.
C will contain the value of the MSB before the operation.

Example: If the accumulator contains the binary value 10111001 then the instruction

COM A

will cause the accumulator to be changed to 01000110 and the carry flag to be set (since the original MSB was 1).

Addressing Modes: Inherent

CP Compare CP

Mnemonic: CP

Function: Compare

Description: This instruction compares the source byte (subtracted from) with the destination byte, which must be the accumulator. The carry and zero flags record the result of this comparison.

Operation: dst - src

The destination must be the accumulator, but it will not be changed.

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
CP dst,src				Z	C
CP A,A	3F FF	2	4	Δ	Δ
CP A,X	3F 80	2	4	Δ	Δ
CP A,Y	3F 81	2	4	Δ	Δ
CP A,V	3F 82	2	4	Δ	Δ
CP A,W	3F 83	2	4	Δ	Δ
CP A,(X)	27	1	4	Δ	Δ
CP A,(Y)	2F	1	4	Δ	Δ
CP A,rr	3F rr	2	4	Δ	Δ

Note:
rr. 1 Byte dataspace address

ST60 Δ: Z is set if the result is zero. Cleared otherwise.

C is set if Acc ≥ src, cleared if Acc < src.

ST62/63 Δ: Z is set if the result is zero. Cleared otherwise.

C is set if Acc < src, cleared if Acc ≥ src.

Example: If the accumulator contains the value 11111000 and the register 34h contains the value 00011100 then the instruction,

CP A,34h

will clear the Zero flag Z and set the Carry flag C, indicating that Acc ≥ src (on ST60)

Addressing Modes: Source: Direct, Indirect

Destination: Accumulator

CPI — Compare Immediate — CPI

Mnemonic: CPI

Function: Compare Immediate

Description: This instruction compares the immediately addressed source byte (subtracted from) with the destination byte, which must be the accumulator. The carry and zero flags record the result of this comparison.

Operation: dst - src

The source must be the immediately addressed data and the destination must be the accumulator, that will not be changed.

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
CPI dst,src				Z	C
CPI A,nn	37 nn	2	4	Δ	Δ

Note:
nn. 1 Byte immediate data.

ST60 Δ: Z is set if the result is zero. Cleared otherwise.

C is set if Acc ≥ src, cleared if Acc < src.

ST62/63 Δ: Z is set if the result is zero. Cleared otherwise.

C is set if Acc < src, cleared if Acc ≥ src.

Example: If the accumulator contains the value 11111000 then the instruction,

CPI A,00011100B

will clear the Zero flag Z and set the Carry flag C indicating that Acc ≥ src (on ST60).

Addressing Modes: Source: Immediate

Destination: Accumulator

DEC **Decrement** DEC

Mnemonic: DEC

Function: Decrement

Description: The destination register's contents are decremented by one.

Operation: dst ← dst - 1

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
DEC dst				**Z**	**C**
DEC A	FF FF	2	4	Δ	*
DEC X	1D	1	4	Δ	*
DEC Y	5D	1	4	Δ	*
DEC V	9D	1	4	Δ	*
DEC W	DD	1	4	Δ	*
DEC (X)	E7	1	4	Δ	*
DEC (Y)	EF	1	4	Δ	*
DEC rr	FF rr	2	4	Δ	*

Notes:

rr. 1 Byte dataspace address

*. C is unaffected

Δ. Z is set if the result is zero. Cleared otherwise.

Example: If the X register contains the value 45h and the data space register 45h contains the value 16h then the instruction,

DEC (X)

will cause data space register 45h to contain the value 15h.

Addressing Modes: Short direct, Direct, Indirect.

INC — Increment — INC

Mnemonic: INC

Function: Increment

Description: The destination register's contents are incremented by one.

Operation: dst ← dst + 1

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
INC dst				Z	C
INC A	7F FF	2	4	Δ	*
INC X	15	1	4	Δ	*
INC Y	55	1	4	Δ	*
INC V	95	1	4	Δ	*
INC W	D5	1	4	Δ	*
INC (X)	67	1	4	Δ	*
INC (Y)	6F	1	4	Δ	*
INC rr	7F rr	2	4	Δ	*

Notes:

rr. 1 Byte dataspace address

*. C is unaffected

Δ. Z is set if the result is zero. Cleared otherwise.

Example: If the X register contains the value 45h and the data space register 45h contains the value 16h then the instruction

INC (X)

will cause data space register 45h to contain the value 17h.

Addressing Modes: Short direct, Direct, Indirect.

JP Jump JP

Mnemonic: JP

Function: Jump (Unconditional)

Description: The JP instruction replaces the PC value with a 12-bit value thus causing a simple jump to another location in the program memory. The previous PC value is lost, not stacked.

Operation: PC ← dst

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
JP dst				**Z**	**C**
JP abc	c1001 ab	2	4	*	*

Notes:
abc. the three half bytes of a 12-bit address.
*. C,Z not affected

Example: The instruction,

JP 5CDh

will cause the PC to be loaded with 5CDh and the program will continue from that location.

Addressing Modes: Extended

JRC Jump Relative on Carry Flag JRC

Mnemonic: JRC

Function: Jump Relative on Carry Flag

Description: This instruction causes the carry (C) flag to be tested and if this flag is set then a jump is performed within the program memory. This jump is in the range -15 to +16 and is relative to the PC value. The displacement e is of five bits. If C=0 then the next instruction is executed.

Operation: If C=1, PC ← PC + e

where e= 5-bit displacement

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
				Z	C
JRC e	e110	1	2	*	*

Notes:

e. 5-bit displacement in the range -15 to +16

*. C,Z not affected

Example: If the carry flag is set then the instruction,

JRC $+8

will cause a branch forward to PC+8. The user can use labels as identifiers and the assembler will automatically allow the jump if it is in the range -15 to +16.

Addressing Modes: Program Counter Relative

JRNC — Jump Relative on Non Carry Flag — JRNC

Mnemonic: JRNC

Function: Jump Relative on Non Carry Flag

Description: This instruction causes the carry (C) flag to be tested and if this flag is cleared to zero then a jump is performed within the program memory. This jump is in the range -15 to +16 and is relative to the PC value. The displacement is of five bits. If C=1 then the next instruction is executed.

Operation: If C=0, PC 🡐 PC + e

where e= 5-bit displacement

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
				Z	C
JRNC e	e010	1	2	*	*

Notes:

e: 5-bit displacement in the range -15 to +16

*: C,Z not affected

Example: If the carry flag is cleared then the instruction,

JRNC $-5

will cause a branch backward to PC-5. The user can use labels as identifiers and the assembler will automatically allow the jump if it is in the range -15 to +16.

Addressing Modes: Program Counter Relative

JRNZ — Jump Relative on Non Zero Flag — JRNZ

Mnemonic: JRNZ

Function: Jump Relative on Non Zero Flag

Description: This instruction causes the zero (Z) flag to be tested and if this flag is cleared to zero then a jump is performed within the program memory. This jump is in the range -15 to +16 and is relative to the PC value. The displacement is of five bits. If Z=1 then the next instruction is executed.

Operation: If Z=0, PC ← PC + e

where e= 5-bit displacement

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
				Z	C
JRNZ e	e000	1	2	*	*

Notes:

e. 5-bit displacement in the range -15 to +16.

*. C,Z not affected

Example: If the zero flag is cleared then the instruction,

JRNZ $-5

will cause a branch backward to PC-5. The user can use labels as identifiers and the assembler will automatically allow the jump if it is in the range -15 to +16.

Addressing Modes: Program Counter Relative

JRR Jump Relative if Reset JRR

Mnemonic: JRR

Function: Jump Relative if RESET

Description: This instruction causes a specified bit in a given dataspace register to be tested. If this bit is reset (=0) then the PC value will be changed and a relative jump will be performed within the program. The relative jump range is -126 to +129. If the tested bit is not reset then the next instruction is executed.

Operation: If bit=0, PC ← PC + ee

where ee= 8-bit displacement

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
				Z	C
JRR b,rr,ee	b00011 rr ee	3	5	*	Δ

Notes:

b. 3-bit address

rr. 1 Byte dataspace address

ee. 8-bit displacement in the range -126 to +129

*. Z is not affected

Δ. The tested bit is shifted into carry.

Example: If bit 4 of dataspace register 70h is reset and the PC=110 then the instruction,

JRR 4, 70h, $-20

will cause the PC to be changed to 90 (110-20) and the instruction starting at that address in the program memory to be the next instruction executed.

The user is advised to use labels for conditional jumps. The relative jump will be calculated by the assembler. The jump must be in the range -126 to +129.

Addressing Modes: Bit Test

JRS — Jump Relative if Set — JRS

Mnemonic: JRS

Function: Jump Relative if set

Description: This instruction causes a specified bit in a given dataspace register to be tested. If this bit is set (=1) then the PC value will be changed and a relative jump will be performed within the program. The relative jump range is -126 to +129. If the tested bit is not set then the next instruction is executed.

Operation: If bit=1, PC 🡐 PC + ee

where ee= 8-bit displacement

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
				Z	C
JRS b,rr,ee	b10011 rr ee	3	5	*	∆

Notes:

b. 3-bit address

rr. 1 Byte dataspace address

ee. 8-bit displacement in the range -126 to +129

*. Z is not affected

∆. The tested bit is shifted into carry.

Example: If bit 7 of dataspace register AFh is set and the PC=123 then the instruction,

JRS 7,AFh,$+25

will cause the PC to be changed to 148 (123+25) and the instruction starting at that address in the program memory to be the next instruction executed.

The user is advised to use labels for conditional jumps. The relative jump will be calculated by the assembler. The jump must be in the range -126 to +129.

Addressing Modes: Bit Test

JRZ — Jump Relative on Zero Flag — JRZ

Mnemonic: JRZ

Function: Jump Relative on Zero Flag

Description: This instruction causes the zero (Z) flag to be tested and if this flag is set to one then a jump is performed within the program memory. This jump is in the range -15 to +16 and is relative to the PC value. The displacement is of five bits. If Z=0 then next instruction is executed.

Operation: If Z=1, PC ← PC + e

where e= 5-bit displacement

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
				Z	C
JRZ e	e100	1	2	*	*

Notes:

e. 5-bit displacement in the range -15 to +16.

*. C,Z not affected

Example: If the zero flag is set then the instruction,

JRZ $+8

will cause a branch forward to PC+8. The user can use labels as identifiers and the assembler will automatically allow the jump if it is in the range -15 to +16.

Addressing Modes: Program Counter Relative

LD — Load — LD

Mnemonic: LD

Function: Load

Description: The contents of the source register are loaded into the destination register. The source register remains unaltered and the previous contents of the destination register are lost.

Operation: dst ← src

Either the source or the destination must be the accumulator.

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
LD dst,src				Z	C
LD A,X	35	1	4	Δ	*
LD A,Y	75	1	4	Δ	*
LD A,V	B5	1	4	Δ	*
LD A,W	F5	1	4	Δ	*
LD X,A	3D	1	4	Δ	*
LD Y,A	7D	1	4	Δ	*
LD V,A	BD	1	4	Δ	*
LD W,A	FD	1	4	Δ	*
LD A,(X)	07	1	4	Δ	*
LD (X), A	87	1	4	Δ	*
LD A,(Y)	0F	1	4	Δ	*
LD (Y),A	8F	1	4	Δ	*
LD A,rr	1F rr	2	4	Δ	*
LD rr,A	9F rr	2	4	Δ	*

Notes:

rr. 1 Byte dataspace address

*. C not affected

Δ. Z is set if the result is zero. Cleared otherwise.

Example: If data space register 34h contains the value 45h then the instruction;

LD A,34h

will cause the accumulator to be loaded with the value 45h. Register 34h will keep the value 45h.

Addressing Modes: Source: Direct, Short Direct, Indirect

Destination: Direct, Short Direct, Indirect

LDI — Load Immediate — LDI

Mnemonic: LDI

Function: Load Immediate

Description: The immediately addressed data (source) is loaded into the destination data space register.

Operation: dst ← src

The source is always an immediate data while the destination can be the accumulator, one of the X,Y,V,W registers or one of the available data space registers.

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
LDI dst,src				Z	C
LDI A,nn	17 nn	2	4	Δ	*
LDI X,nn	0D 80 nn	3	4	*	*
LDI Y,nn	0D 81 nn	3	4	*	*
LDI V,nn	0D 82 nn	3	4	*	*
LDI W,nn	0D 83 nn	3	4	*	*
LDI rr,nn	0D rr nn	3	4	*	*

Notes:

rr. 1 Byte dataspace address

nn. 1 Byte immediate value

*. Z, C not affected

Δ. Z is set if the result is zero. Cleared otherwise.

Example: The instruction

LDI 34h, 45h

will cause the value 45h to be loaded into data register at location 34h.

Addressing Modes: Source: Immediate

Destination: Direct

NOP No Operation NOP

Mnemonic: NOP

Function: No Operation

Description: No action is performed by this instruction. It is typically used for timing delay.

Operation: No Operation

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
				Z	C
NOP	04	1	2	*	*

Note:
*. C,Z not affected

Addressing Modes: Program Counter Relative

RES Reset Bit RES

Mnemonic: RES

Function: Reset Bit

Description: The RESET instruction is used to reset a specified bit in a given register in the dataspace.

Operation: dst (n) 0, $0 \leq n \leq 7$

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
RES bit,dst				Z	C
RES b,A	b01011 FF	2	4	*	*
RES b,rr	b01011 rr	2	4	*	*

Notes:

b. 3-bit address

rr. 1 Byte dataspace address

*. C,Z not affected

Example: If register 23h of the dataspace contains 11111111 then the instruction,

RES 4,23h

will cause register 23h to hold 11101111.

Addressing Modes: Bit Direct

RET — Return from Subroutine — RET

Mnemonic: RET

Function: Return From Subroutine

Description: This instruction is normally used at the end of a subroutine to return to the previously executed procedure. The previously stacked program counter (stacked during CALL) is popped back from the stack. The next statement executed is that addressed by the new contents of the PC. If the stack had already reached its highest level (no more PC stacked) before the RET is executed, program execution will be continued at the next instruction after the RET.

Operation: PC ← Stacked PC

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
				Z	C
RET	CD	1	2	*	*

Note:
*. C,Z not affected

Example: If the current PC value is 456h and the PC value at the top of the stack is 3DFh then the instruction,

RET

will cause the PC value 456h to be lost and the current PC value to be 3DFh.

Addressing Modes: Inherent

RETI Return from Interrupt RETI

Mnemonic: RETI

Function: Return from Interrupt

Description: This instruction marks the end of the interrupt service routine and returns the ST60/62/63 to the state it was in before the interrupt. It "pops" the top (last in) PC value from the stack into the current PC. This instruction also causes the ST60/62/63 to switch from the interrupt flags to the normal flags. The RETI instruction also applies to the end of NMI routine for ST62/63 devices; in this case the instruction causes the switch from NMI flags to normal flags (if NMI was acknowledged inside a normal routine) or to standard interrupt flags (if NMI was acknowledged inside a standard interrupt service routine).

In addition the RETI instruction also clears the interrupt mask (also NMI mask for ST62/63) which was set when the interrupt occurred. If the stack had already reached its highest level (no more PC stacked) before the RETI is executed, program execution will be continued with the next instruction after the RETI. Because the ST60 is in interrupt mode after reset (NMI mode for ST62/63), RETI has to be executed to switch to normal flags and enable interrupts at the end of the starting routine. If no call was executed during the starting routine, program execution will continue with the instruction after the RETI (supposed no interrupt is active).

Operation: Actual Flags Normal Flags (1)

PC 🡐 Stacked PC

IM 🡐 0

(1) Standard Interrupt flags if NMI was acknowledged inside a standard interrupt service (ST62/63 only).

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
				Z	C
RETI	4D	1	2	Δ	Δ

Note:

Δ C,Z normal flag will be used from now on.

Example: If the current PC value is 456h and the PC value at the top of the stack is 3DFh then the instruction

RETI

will cause the value 456h to be lost and the current PC value to be 3DFh. The ST6 will switch from interrupt flags to normal flags and the interrupt mask is cleared.

Addressing Modes: Inherent

RLC

Rotate Left Through Carry RLC

Mnemonic: RLC

Function: Rotate Left through Carry

Description: This instruction moves each bit in the accumulator one place to the left (i.e. towards the MSBit. The MSBit (bit 7) is moved into the carry flag and the carry 00000000000000 flag is moved into the LSBit (bit0) of the accumulator.

Operation:

dst(0) 🡐 C

C 🡐 dst(7)

dst(n+1) 🡐 dst(n), $0 \leq n \leq 6$

This instruction can only be performed on the accumulator.

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
				Z	C
RLC A	AD	1	4	Δ	Δ

Note:

Δ: Z is set if the result is zero. Cleared otherwise.
C will contain the value of the MSB before the operation.

Example: If the accumulator contains the binary value 10001001 and the carry flag is set to 0 then the instruction,

RLC A

will cause the accumulator to have the binary value 00010010 and the carry flag to be set to 1.

Addressing Modes: Inherent

SET — Set Bit — SET

Mnemonic: SET

Function: Set Bit

Description: The SET instruction is used to set a specified bit in a given register in the data space.

Operation: dst (n) ← 1, $0 \le n \le 7$

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
SET bit,dst				Z	C
SET b,A	b11011 FF	2	4	*	*
SET b,rr	b11011 rr	2	4	*	*

Notes:

b. 3-bit address

rr. 1 Byte dataspace address

*. C,Z not affected

Example: If register 23h of the dataspace contains 00000000 then the instruction,

SET 4,23h

will cause register 23h to hold 00010000.

Addressing Modes: Bit Direct

SLA Shift Left Accumulator SLA

Mnemonic: SLA

Function: Shift Left Accumulator

Description: This instruction implements an addition of the accumulator to itself (i.e a doubling of the accumulator) causing an arithmetic left shift of the value in the register.

Operation: ADD A,FFh

This instruction can only be performed on the accumulator.

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
				Z	C
SLA A	5F FF	2	4	Δ	Δ

Note:

Δ: Z is set if the result is zero. Cleared otherwise.
C will contain the value of the MSB before the operation.

Example: If the accumulator contains the binary value 11001101 then the instruction,

SLA A

will cause the accumulator to have the binary value 10011010 and the carry flag to be set to 1.

Addressing Modes: Inherent

STOP **Stop Operation** STOP

Mnemonic: STOP

Function: Stop operation

Description: This instruction is used for putting the ST60/62/63 into a stand-by mode in which the power consumption is reduced to a minimum. All the on-chip peripherals and oscillator are stopped (for some peripherals, A/D for example, it is necessary to individually turn-off the macrocell before entering the STOP instruction). To restart the processor an external interrupt or a reset is needed.

Operation: Stop Processor

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
				Z	C
STOP	6D	1	2	*	*

Note:
*: C,Z not affected

Addressing Mode: Inherent

SUB Subtraction SUB

Mnemonic: SUB

Function: Subtraction

Description: This instruction subtracts the source value from the destination value.

Operation: dst ← dst-src

The destination must be the accumulator.

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
SUB dst,src				Z	C
SUB A,A	DF FF	2	4	Δ	Δ
SUB A,X	DF 80	2	4	Δ	Δ
SUB A,Y	DF 81	2	4	Δ	Δ
SUB A,V	DF 82	2	4	Δ	Δ
SUB A,W	DF 83	2	4	Δ	Δ
SUB A,(X)	C7	1	4	Δ	Δ
SUB A,(Y)	CF	1	4	Δ	Δ
SUB A,rr	DF rr	2	4	Δ	Δ

Note:
rr. 1 Byte dataspace address

ST60 Δ: Z is set if the result is zero. Cleared otherwise.
C is set if Acc ≥ src, cleared if Acc < src.

ST62/63 Δ: Z is set if the result is zero. Cleared otherwise.
C is set if Acc < src, cleared if Acc ≥ src.

Example: If the Y register contains the value 23h, dataspace register 23h contains the value 53h and the accumulator contains the value 78h then the instruction,

SUB A,(Y)

will cause the accumulator to hold the value 25h (i.e. 78-53). The zero flag is cleared and the carry flag is set (on ST60), indicating that result is > 0.

Addressing Modes: Source: Indirect,Direct

Destination: Accumulator

SUBI — Subtraction Immediate — SUBI

Mnemonic: SUBI

Function: Subtraction Immediate

Description: This instruction causes the immediately addressed source data to be subtracted from the accumulator.

Operation: dst ← dst - src

The destination must be the accumulator.

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
SUBI dst,src				Z	C
SUBI A,nn	D7 nn	2	4	Δ	Δ

Note:
nn. 1 Byte of immediate data

ST60 Δ: Z is set if the result is zero. Cleared otherwise.
C is set if Acc ≥ src, cleared if Acc < src.

ST62/63 Δ: Z is set if the result is zero. Cleared otherwise.
C is set if Acc < src, cleared if Acc ≥ src.

Example: If the accumulator contains the value 56h then the instruction,

SUBI A,25

will cause the accumulator to contain the value 31h. The zero flag is cleared and the carry flag is set (on ST60), indicating that the result is > 0.

Addressing Modes: Source: Immediate

Destination: Accumulator

WAIT Wait Processor WAIT

Mnemonic: WAIT

Function: Wait Processor

Description: This instruction is used for putting the ST60/62/63 into a stand-by mode in which the power consumption is reduced to a minimum. Instruction execution is stopped, but the oscillator and some on-chip peripherals continue to work. To restart the processor an interrupt from an active on-chip peripheral (e.g. timer), an external interrupt or reset is needed. For on-chip peripherals active during wait, see ST60/62/63 data sheets.

Operation: Put ST6 in stand-by mode

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
				Z	C
WAIT	ED	1	2	*	*

Note:
*. C,Z not affected

Addressing Modes: Inherent

Notes