# ESR-L INSTRUCTION MANUAL

for PC-E500
CE-7000 series
EG-7000 series

SHARP CORPORATION

## General Specifications

- 8-bit C-MOS CPU
- Memory space (1 Mbyte)
  - Data: 1 Mbyte continuous
  - Program: 64 Kbytes x 16 segments
- Built-in 236-byte RAM
  - This internal memory makes up a separate memory space from the external memory, and its address allocation is as follows:
    - 00H to EBH: RAM
    - ECH to FFH: Pointers, control command registers, data buffers, I/O ports, etc.
- Main clock
  - Ceramic-oscillated 2.304 MHz (1.30 μsec. execute cycle)
  - Changeable by PLA to:
    - 1.536 MHz (1.95 μsec. execute cycle), or
    - 3.072 MHz (0.98 μsec. execute cycle)
- Sub-clock
  - CR-oscillated typ. 40 kHz (As a display clock and as a timer for a half), or
  - Crystal-oscillated 32.768 kHz (Used when providing a clock function).
- Interrupt
  - 7 factors: 2 timers, 2 keys, 2 UARTs, and external factor.
- UART

| | |
|---|---|
| Baud rate: | 300 to 9600 |
| Parity: | Even/odd/no |
| Character length: | 7/8 |
| Stop bit: | 1/2 |

  Breakout function

- I/O

| | |
|---|---|
| A0 to A18: | Address buses |
| CE0 to CE7: | Chip select output pins |
| DIO0 to DIO7: | Data buses |
| MRQ RD WR: | Memory control pin |

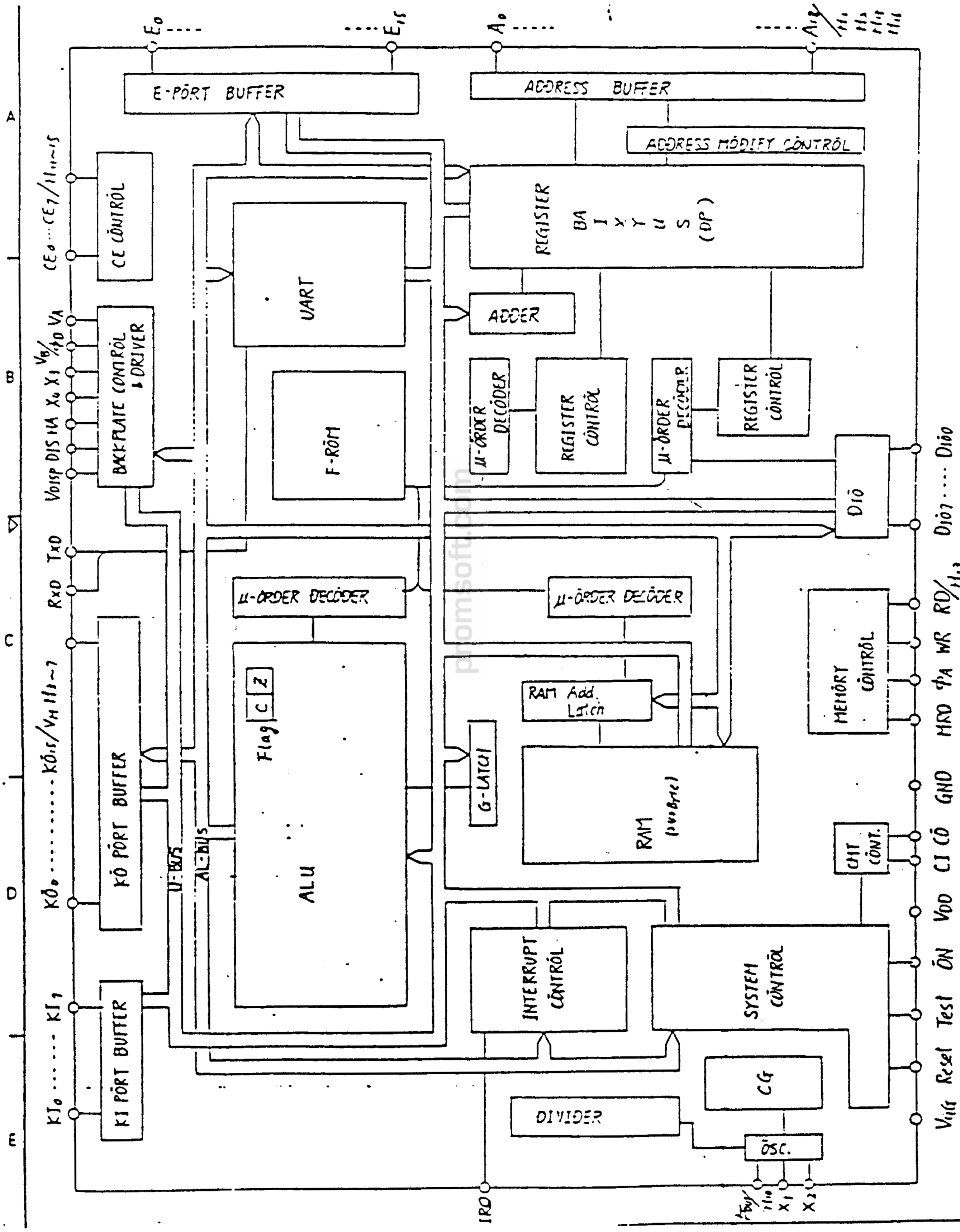
| | |
|---|---|
| KO0 to KO15: | Key strobe output ports |
| KI0 to KI7: | Key input ports |
| E0 to E15: | General I/O ports |
| øA HA DIS: | LCD driver control pin |
| TXD: | UART data output pin |
| RXD: | UART data input pin |
| CO: | CMT data output pin |
| CI: | CMT data input pin |
| IRQ: | External interrupt input pin |
| ON: | ON key input pin |

. LCD common signal output allowed — 1/16 duty. However, the number of pins is only 14, from H1 to H15 excluding H8. Changed with A, CE, KO, and other pins by PLA.

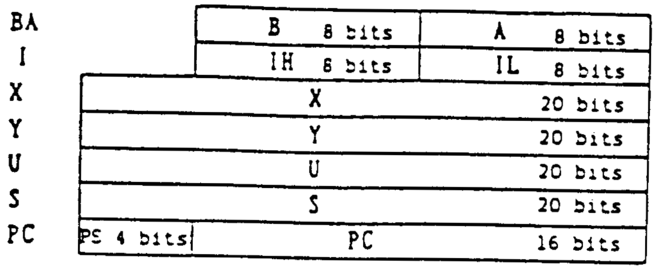
. HALT/OFF function — HALT: Stops the main clock.

OFF : Stops both the main clock and the sub-clock.

Block Diagram



Registers



A Accumulator: An arithmetic register. The ESR-L performs arithmetic mainly on the accumulator and the internal memory.

B Auxiliary register: An auxiliary register for the accumulator. Used as a 16-bit register in combination with the register A.

I Counter register: An loop instruction automatically decrements this register till the data it has beccmes zero.

X Pointer register: Allowed to directly access addresses throughout the 1-megabyte external memory.

Y Pointer register: Allowed to directly access addresses throughout the 1-megabyte external memory.

U User stack pointer: A stack pointer that is used by the user to save data or for data transfer between routines. This stack pointer is also usable as a pointer register like the registers X and Y, and is allowed to directly access the 1-megabyte space.

S System stack pointer: A stack pointer that is mainly used by the system for subroutine call, etc. It is also usable as a pointer register like the registers X and Y, and is allowed to directly access to 1-megabyte space.

PC Program counter: Automatically increments every time an object code is fetched, i.e. every 16 bits. Therefore, 64 K bytes of object codes makes 1 segment.

PS Page segment register: Specifies a page from which an object code is to be fetched. This register consists of 4 bits, and segments are from 0H to FH.

## Memory Space

### Internal memory space (256 bytes)

| Address | | |
|---|---|---|
| 00H | Internal memory 236 bytes | |
| EBH | | |
| ECH | BP | RAM BASE POINTER |
| EDH | PX | RAM PX POINTER |
| EEH | PY | RAM PY POINTER |
| EFH | AMC | ADR.MODIFY CONTROL |
| F0H | KOL | KEY OUTPUT BUFFER |
| F1H | KOH | KEY OUTPUT BUFFER |
| F2H | KI | KEY INPUT BUFFER |
| F3H | EOL | E PORT OUTPUT BUFFER |
| F4H | EOH | E PORT OUTPUT BUFFER |
| F5H | EIL | E PORT INPUT BUFFER |
| F6H | EIL | E PORT INPUT BUFFER |
| F7H | UCR | UART CONTROL REGISTER |
| F8H | USR | UART STATUS REGISTER |
| F9H | RXD | UART RECEIVE BUFFER |
| FAH | TXD | UART TRANSMIT BUFFER |
| FBH | IMR | INTERRUPT MASK REGISTER |
| FCH | ISR | INTERRUPT STATUS REGISTER |
| FDH | SCR | SYSTEM CONTROL REGISTER |
| FEH | LCC | LCD CONTRAST CONTROL |
| FFH | SSR | SYSTEM STATUS REGISTER |

### External memory space (1 Mbyte)

| Address | | |
|---|---|---|
| 00000H | | SH-26 area |
| 03FFFH | | |
| 04000H | | LH-5073A1 area |
| 07FFFH | | |
| 08000H | 1 Mbyte | |
| FFFFAH | Low | INTERRUPT VECTOR |
| FFFFBH | High | |
| FFFFCH | Ext. | |
| FFFFDH | Low | RESET VECTOR |
| FFFFEH | High | |
| FFFFFH | Ext. | |

## Internal Memory Space

The internal memory space has 256 bytes and is accessed by the 8-bit pointer BP, PX, or PY or an immediate value. It makes up a difference space from the external memory space. Among the 256 bytes, 236 bytes from 00H to EBH make ordinary RAM, and the rest, i.e. 20 bytes from ECH to FFH, is dedicated to pointers, control command registers, data buffers, and I/O ports.

RAM Addresses: 00H to EBH Read/Write

These 236 bytes make ordinary RAM.

BP Address: ECH Read/Write

RAM base pointer: A pointer to specify the reference address for accessing an address in the internal memory space by relative addressing. The following are the addressing modes that use this pointer:

(bp±n) .... In this mode, an effective address is obtained by adding the offset value "±n" to the information the BP has.

(bp+px) ... In this mode, an effective address is obtained by adding the data the PX has to the data the BP has.

(bp+py) ... In this mode, an effective address is obtained by adding the data the PY has to the data the BP has.

PX Address: EDH Read/Write

RAM PX pointer: A pointer that specifies an address on the internal memory and is used when data on the internal memory is subjected to loop processing. There are the following two addressing modes that use the RAM PX pointer:

(px±n) .... An effective address is obtained by adding the offset value "±n" to the data the PX has.

(bp+px) ... An effective address is obtained by adding the data the PX has to the data BP has.

PY Address: EEH Read/Write

RAM PY pointer: A pointer that specifies an address on the internal memory as the PX does. The PY is used when the second operand specifies a pointer in an instruction both of whose first and second operands use internal memory addressing.

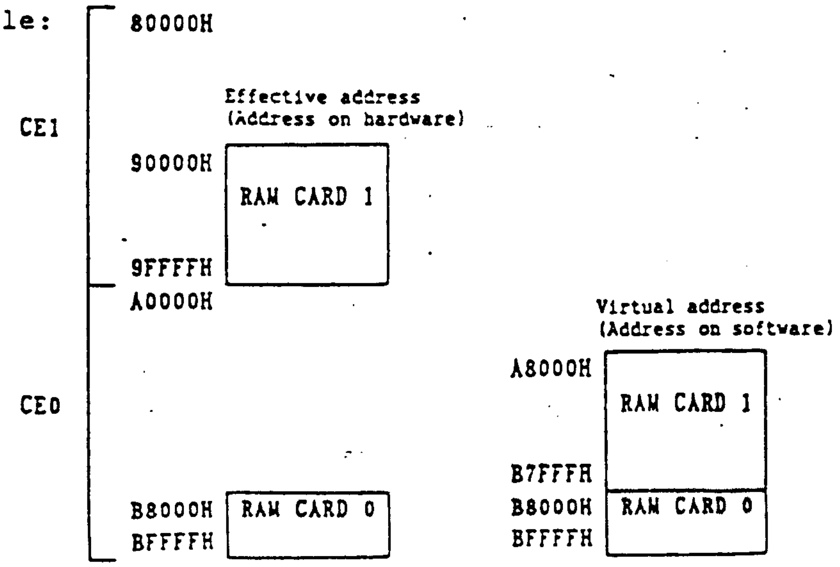
(py±n) .... This addressing mode obtains an effective address by adding the offset value "±n" to the data the PY has.

(bp±py) ... This addressing mode obtains an effective address by adding the data the PY has to the data the BP has.

AMC Address: EFH Read/Write

Address Modify Control: A system provided with two RAM card slots may have two discontinuous RAM card address areas depending on the capacities of the RAM cards inserted. This register controls the function that virtually makes the RAM card address areas continuous. When the function is enabled, it works between the CE1 and the CE2 and allows the system to use the address areas with the end of the CE1 address area virtually linked to the start address of the CE0 address area.

Example:



| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| AME | | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |

AME ....... Address Modify Enable: Enables the address modify function when this bit is "1".

AM0 to
AM5 ....... These specify the RAM card capacity at the CE0 side:

| AM5 | AM4 | AM3 | AM2 | AM1 | AM0 | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | ... | 2 Kbytes |
| 0 | 0 | 0 | 0 | 0 | 1 | ... | 4 Kbytes |
| 0 | 0 | 0 | 0 | 1 | 1 | ... | 8 Kbytes |
| 0 | 0 | 0 | 1 | 1 | 1 | ... | 16 Kbytes |
| 0 | 0 | 1 | 1 | 1 | 1 | ... | 32 Kbytes |
| 0 | 1 | 1 | 1 | 1 | 1 | ... | 64 Kbytes |
| 1 | 1 | 1 | 1 | 1 | 1 | ... | 128 Kbytes |

The system that uses the address modify function is allowed to set the access area of the CE0 and CE1 to 32, 64, 128, or 256 K bytes by the PLA. The largest numbered address of the CE0 must be XFFFFH.

KOL Address: F0H Read/Write

Key Output Buffer: The data stored in this buffer is outputted as it is through KO port (This port is mainly for key strobe).

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| KO7 | KO6 | KO5 | KO4 | KO3 | KO2 | KO1 | KO0 |

KO0 to KO7 ... These correspond respectively to Pins KO0 to KO7. A "high" signal is outputted when the bit is "1", and a "low" signal when "0".

KOH Address: F1H Read/Write

Key Output Buffer: The data stored in this buffer is outputted as it is through the KO port (This port is mainly for key strobe).

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| KO15 | KO14 | KO13 | KO12 | KO11 | KO10 | KO9 | KO8 |

KO8 to KO15 .. These correspond respectively to Pins KO8 to KO15. A "high" signal is outputted when the bit is "1", and a "low" signal when "0".

KI Address: F2H Read only

Key Input Buffer: This buffer stores the information that shows the status of the KI port (mainly for key input).

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| KI7 | KI6 | KI5 | KI4 | KI3 | KI2 | KI1 | KI0 |

KIO to KI7 ... These correspond respectively to Pins KIO to KI7. Bit "1" is inputted when the level is "high", and bit "0" when "low".

EOL Address: F3H Read/Write

E Port Output Buffer: The data stored in this buffer is outputted as it is through the E port (general I/O port).

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| E07 | E06 | E05 | E04 | E03 | E02 | E01 | E00 |

E0 to E7 ..... These correspond respectively to Pins E0 to E7. A "high" signal is outputted when the bit is "1", and a "low" signal when "0".

EOH Address: F4H Read/Write

E Port Output Buffer: The data stored in this buffer is outputted as it is through the E port (general I/O port).

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| E015 | E014 | E013 | E012 | E011 | E010 | E09 | E08 |

E08 to E15 ... These correspond respectively to Pins E8 to E15. A "high" signal is outputted when the bit is "1", and a "low" signal when "0".

EIL Address: F5H Read only

E Port Input Buffer: This buffer stores the information that shows the status of the E port (general I/O port).

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EI7 | EI6 | EI5 | EI4 | EI3 | EI2 | EI1 | EI0 |

EI0 to EI7 ... These correspond respectively to Pins E0 to E7. Bit "1" is inputted when the level is "high", and bit "0" when "low".

EIH Address: F6H Read only

E Port Input Buffer: This buffer stores the, information that shows the status of the E port (general I/O port).

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EI15 | EI14 | EI13 | EI12 | EI11 | EI10 | EI9 | EI8 |

EI8 to EI15 .. These correspond respectively to Pins E8 to E15. Bit "1" is inputted when the level is "high", and bit "0" when "low"..

UCR Address: F7H Read/Write

UART Control Register: This register is used to set the status of the UART.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| BOE | BR2 | BR1 | BR0 | PA1 | PA0 | DL | ST |

BOE .......... Break Output Enable: While this bit is kept to "1", "0" is outputted through Pin TXD.

BR0 to BR2 ... Baud Rate Factor: By these bits, the baud rate is set.

| BR2 | BR1 | BR0 | | BPS | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | ··· | 0 | (This resets the UART.) |
| 0 | 0 | 1 | ··· | 300 | |
| 0 | 1 | 0 | ··· | 600 | |
| 0 | 1 | 1 | ··· | 1200 | |
| 1 | 0 | 0 | ··· | 2400 | |
| 1 | 0 | 1 | ··· | 4800 | |
| 1 | 1 | 0 | ··· | 9600 | |
| 1 | 1 | 1 | ··· | 19200 | |

PA0 & PA1 .... Parity: This bits determine the parity.

| PA1 | PA0 | | PARITY |
|---|---|---|---|
| 0 | 0 | ··· | EVEN |
| 0 | 1 | ··· | ODD |
| 1 | X | ··· | NO |

DL ........... Character Length: This bit determines the data length.

| DL | | bit |
|---|---|---|
| 0 | ··· | 8 |
| 1 | ··· | 7 |

ST ........... Stop Bit: This determines the stop bit length.

| ST | | bit |
|---|---|---|
| 0 | ··· | 1 |
| 1 | ··· | 2 |

USR Address: F8H Read only

UART Status Register: A register for checking the UART status.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | RXR | TXE | TXR | FE | OE | PE |

RXR .......... Receiver Ready: This bit assumes "1" when 1 character receiving is completed, and it becomes "0" when the data stored in the UART receive buffer (F9H) is transferred to a register (BA or other).

TXE .......... Transmitter Empty: This bit is "0" while the transmitter of the UART is sending data. Otherwise, it is "1".

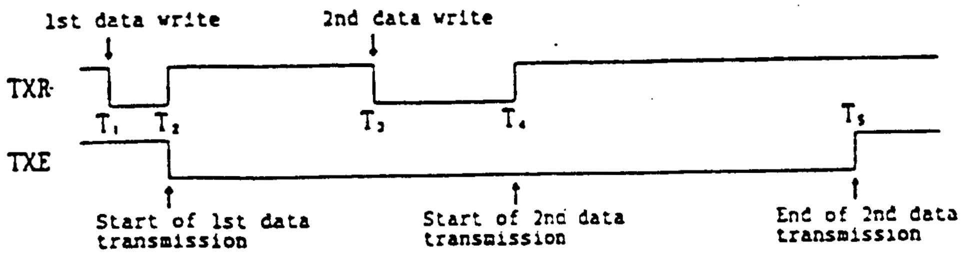
TXR .......... Transmitter Ready: This bit becomes "0" when data is written in the UART transmit buffer (FAH), and it changes to "1" when that data is delivered to the UART transmitter.

FE ........... Framing Error: This bit becmes "0" if "1" is received just when the stop bit ought to arrive. Otherwise, it is set to "1". The value is updated by every completion of 1 character receiving.

OE ........... Overrun Error: This bit becomes "1" when the RXR is "1" at the time of completion of 1 character receiving (i.e. at the time of completing data receiving before the previously received data is read from the UART receive buffer). However, since the OE bit actually becomes "1" a little (i.e. about a half of the time required for 1-bit transfer) before completing 1 character receiving, read the UART status register (F8H) first and the UART receive buffer (F9H) in succession. The OE bit value is updated every time 1 character receiving is completed.

PE ........... Parity Error: This bit is set to "1" upon completion of 1 character receiving if the parity requirement is not met. The value is updated every time 1 character receiving is completed.

Supplement: Relation between data write in the UART transmit buffer (FAH) and TXE and TXR value transition


1st data write
2nd data write
TXR
$T_1$
$T_2$
$T_3$
$T_4$
$T_5$
TXE
Start of 1st data transmission
Start of 2nd data transmission
End of 2nd data transmission

$T_1$: TXR becomes "0" upon the first data write in the UART transmit buffer (FAH). TXE still remains "0" at this moment.

$T_2$: After a lapse of time for a transfer of 1 or 2 bits, the data written is transferred to the transmitter which in turn starts transmission. At that moment, TXR becomes "1" and TXE "0". However, TXR might become "1" 1 state earlier or later than TXE becomes "0". For that reason, avoid checking TXR and TXE simultanecusly.

$T_3$: When the next data is written in the UART transmit buffer while the transmitter is sending the previous data, "0" is set to TXR and the system waits for the end of that data transmission.

$T_4$: The moment the transmitter finishes transmitting the previcus data, the next data written in the UART transmit buffer is transferred to the transmitter, which then starts to transmit the data without any delay. TXR becomes "1" again at that stage.

$T_5$: If there is no more transmit data in the UART transmit buffer when the transmitter finishes the data transmission, TXE becomes "1".

RXD Address: F9H Read only

UART Receive Buffer: Upon completion of 1 character receiving, the data appears in this buffer. When the character length is 7 bits, the MSB is always zero.

TXD Address: FAH Write only

UART Transmit Buffer: The data written in this buffer is transferred tc the UART transmitter when it becomes empty, and is automatically transmitted therefrom after adjusted tc the setting thereof. Data can be written in this buffer even during transmission of the previous data. Whether the previously written data has already been delivered to the transmitter or not yet can be checked by the TXR bit of the UART status register (F8H). If there is data already written in the transmit buffer when the transmission of the previous data is completed, that data is immediately transferred to the transmitter which in turn transmits the data without delay.

IMR Address: FBH Read/Write

Interrupt Mask Register: By writing "0" in a bit of this register, the interrupt by the interrupt factor corresponding to that bit is inhibited. By setting "0" to the MSB, the interrupt by any of the interrupt factors is inhibited. If an interrupt is executed, the data in the interrupt mask register is stored in the system stack and the MSB becomes "0". When the PUSHU IMR command is executed, the data in this buffer is stored in the user stack and the MSB becomes "0".

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IRM | EXM | RXRM | TXRM | ONKM | KEYM | STM | MTM |

IRM .......... Interrupt Mask: If "0" is set to this bit, the interrupt by any of the interrupt factors is inhibited.

If "0" is set to the bit (i.e. any of the bits shown below), the interrupt by the interrupt factor which corresponds to that bit is inhibited. For the meanings of the individual bits, see the descriptions given under "Interrupt Status Register (whose address is FCH)".

EXM ........ External Interrupt Mask
RXRM ....... Receiver Ready Interrupt Mask
TXRM ....... Transmitter Ready Interrupt Mask

ONKM ....... On Key Interrupt Mask
KEYM ....... Key Interrupt Mask
STM ........ SEC Timer Interrupt Mask
MTM ........ MSEC Timer Interrupt Mask

Note: If the AND command is executed on this register when masking any one of the interrupt factors is tried, the interrupt by the factor whose masking has been tried might be caused directly after the AND command execution since the CPU judges whether or not to interrupt the next command before it completes the AND operation. If such interrupt occurs, the CPU cannot determine which factor has caused the interrupt, though the interrupt routine checks the interrupt mask register. Since the factor is already masked. Therefore, it is necessary to let the interrupt routine return without doing anything in case it does not find any interrupt factor to be processed.

ISR Address: FCH Read/Write

Interrupt Status Register: When an interrupt is requested by an interrupt factor, the bit of the interrupt status register which corresponds to that interrupt factor is set to "1". Once "1" is set to the bit, the bit keeps "1" even after the interrupt request disappears unless the software writes "0" in that bit. When a bit of the interrupt status register becomes "1" while both the corresponding bit of the interrupt mask register (FBH) and the MSB are "1", the CPU executes the interrupt before executing the command to be fetched next.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
|  | EXI | RXRI | TXRI | ONKI | KEYI | STI | MTI |

EXI .......... External Interrupt: This bit becomes "1" when an interrupt request comes from an external pin.

RXRI ......... Receiver Ready Interrupt: This bit becomes "1" when the UART completes 1 character receiving.

TXRI ......... Transmitter Ready Interrupt: This bit becomes "1" when the UART transmit buffer (FAH) gets ready for receiving new data after it transfers the data it previously received to the transmitter.

ONKI ......... ON Key Interrupt: This bit becomes "1" when a high" signal is inputted to the ON pin.

KEYI ......... Key Interrupt: This bit becomes "1" when a "high" signal is inputted to any one of the pins of KI (PLA programming can determine which KI pins are effective for this interrupt).

ST1 .:........ SEC Timer Interrupt: This bit becomes "1" when an interrupt is requested by the SUB-CG timer.

MTI .......... MSEC Timer Interrupt: This bit becomes "1" when an interrupt is requested by the Main CG timer.

SCR Address: FDH Read/Write

System Control Register: A register for controling the system status.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ISE | BZ2 | BZ1 | BZ0 | VDDC | STS | MTS | DISC |

ISE .......... IRQ Start Enable: By keeping this bit to "1", the external interrupt is allowed to release the CPU from the HALT/OFF state.

BZ0 to BZ2 ... CO/CI Control Factors: These bits control the state of Pins CO and CI.

| BZ2 | BZ1 | BZ0 | | CO | CI | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | ··· | low | 0 | (Disallows input.) |
| 0 | 0 | 1 | ··· | high | 0 | (Disallows input.) |
| 0 | 1 | 0 | ··· | 2KHZ | 0 | (Disallows input.) |
| 0 | 1 | 1 | ··· | 4KHZ | 0 | (Disallows input.) |
| 1 | 0 | 0 | ··· | low | 0/1 | (Allows input.) |
| 1 | 0 | 1 | ··· | high | 0/1 | (Allows input.) |
| 1 | 1 | X | ··· | CI | 0/1 | (Allows input.) |

VDDC ......... VDD Control: Controls the VDD level.

| VDDC | | VDD |
|---|---|---|
| 0 | ··· | low(VCC) |
| 1 | ··· | high(GND) |

Note: The current hardware (SC62015B01) stops the sub-CG when "1" is set to the VDDC bit. The future hardware will be provided with a PLA for programming whether or not to involve the stop of the sub-CG. When to provide the PLA is, however, to be determined.

STS .......... SEC Timer Select: This bit is used to make a selection between the 2 sub-CG timer working durations: "0" specifies the longer one and "1" the shorter. Since the change of the timer working duration means a change of divider connection, the dividing gets irregular at the instant of the change. For that reason, the change must take place just after the STI becomes "1" or just after the TCL command.

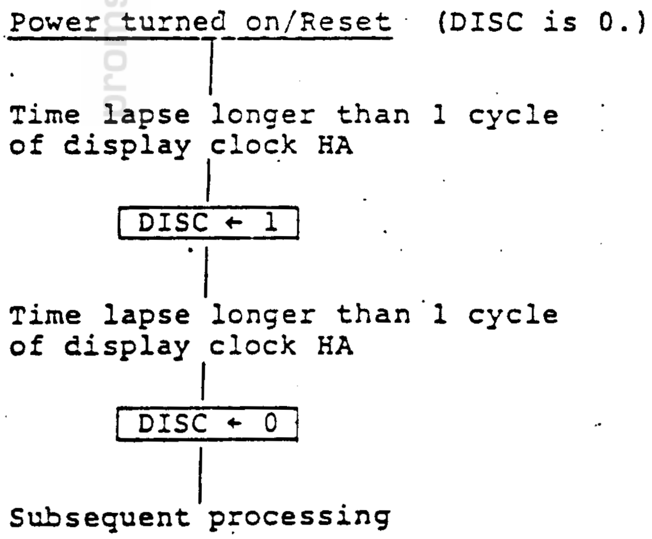
MTS .......... MSEC Timer Select: This bit is used to make a selection between the 2 main CG timer operating durations: "0" specifies the shorter one and "1" the longer. For the same reason as the STS.

The change between them must occur just after the MTI becomes "1" or just after the TCL command.

DISC ......... LCD Driver Control: This bit is used to control the signal level at the DIS pin. Generally, the LCD driver is turned on and off through this pin.

| DISC | | DIS | LCD DRIVER |
|---|---|---|---|
| 0 | ··· | low | DISP OFF |
| 1 | ··· | high | DISP ON |

In addition, the LCD driver display counter is synchronized by raising the DIS signal from the "low" level to the "high". Since the common output signal of the CPU is not synchronized with the LCD segment output signal at the time of ACL, use the DIS signal to synchronize them in the following procedures:


Power turned on/Reset (DISC is 0.)
Time lapse longer than 1 cycle of display clock HA
DISC ← 1
Time lapse longer than 1 cycle of display clock HA
DISC ← 0
Subsequent processing

Moreover, so program the DISC bit as to synchronize them when a key input is made and every 10 seconds.

LCC Address: FEH Read/Write

LCD Contrast Control: Used for LCD display contrast control. It also controls the key strobe disable function and the timer reset function.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| LCC4 | LCC3 | LCC2 | LCC1 | LCC0 | KSD | STCL | MTCL |

LCC0 to LCC4 . LCD Contrast Control Factors: These are used to control the LCD display contrast (32 stages).

| LCC4 | LCC3 | LCC2 | LCC1 | LCC0 | | LCD CONTRAST |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | ... | min. |
| 0 | 0 | 0 | 0 | 1 | | |
| 0 | 0 | 0 | 1 | 0 | | |
| | | : | | | | |
| 1 | 1 | 1 | 1 | 1 | ... | max. |

KSD .......... Key Strobe Disable: While this bit is "1", the signal level is "low" at all the KO pins. The key output buffers (i.e. KOL and KOH) can be used as in normal state.

STCL ......... SEC Timer Clear: Executing the TCL command with this bit set to "1" resets the sub-CG timer.

MTCL ......... MSEC Timer Clear: When the TCL command is executed with this bit set to "1", the main CG timer is reset.

SSR Address: FFH Read only

System Status Register: A register that is used to check the status of the system.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | ONK | RSF | CI | TEST |

ONK .......... ON Key: This bit is "0" when the signal level is "low" at the ON pin, and "1" when "high".

RSF .......... Reset Start Flage: This bit becomes "0" when the level of the signal inputted to the RESET pin is "high", and "1" when the HALT/OFF function is put in action.

CI ........... CMT Input: This bit is "0" when the CI pin is at the "low" signal level, and "1" when at the "high" level.

TEST ......... Test Input: This bit is "0" when the TEST pin is at the "low" signal level, and "1" when at the "high" level.

External Memory Space

The external memory space has a capacity of 1 M byte. Data addresses in the external memory space are specified by using the 20-bit pointer X, Y, U, or S, or in the indexed addressing mode by the internal memory (whose highest-order 4 bits are don't-care bits), or by the 3-byte immediate value (whose highest-order 4 bits are don't-care bits). The 1 M byte is usable as a continuous space.

Program addresses are specified by the 16-bit program counter. The program is therefore divided into 64-kilobyte pages. The page is specified by the 4-bit page segment register.

In the 1-megabyte memory space, the areas from 00000H to 03FFFH and from 04000H to 07FFH are respectively used to connect the SH-26 and the LH-5073A1. The 3 bytes from FFFFAH to FFFFCH are assigned to the interrupt vector, and the 3 bytes from FFFDH to FFFFFH to the reset vector.

SH-26 area Address: 00000H to 03FFFH

This area is provided for LCD driver SH-26 connection. A single access to this area uses 7 states including the time required for address output to the DIO, though a single memory access normally uses 1 state.

New driver area Address: 04000H to 07FFFH

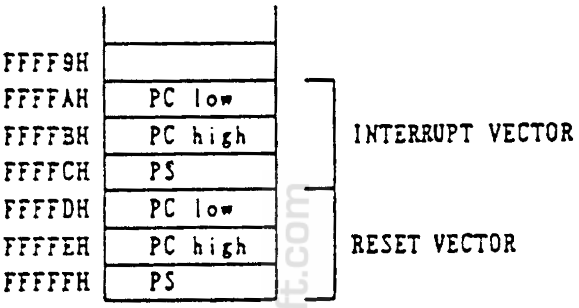
This area is provided for new LCD driver connection. Though a single access to the external memory normally uses 1 state, an access to the new driver area uses 2 states including the time for address output to the DIO in.

Interrupt vector Address: FFFFAH to FFFFCH

The interrupt vector, when an interrupt occurs, specifies the destination to which control jumps after the interruption, i.e. when an interrupt occurs, the 3-byte data stored in the interrupt vector is read into the program counter and program execution restarts therefrom.

Reset vector                    Address:  FFFFDH to FFFFFH

The reset vector specifies the destination to which control jumps after resetting, i.e. when resetting is performed, the 3-byte data in the reset vector is read into the program counter and program execution Starts therefrom.



## ALU

The ALU has arithmetic and logical operation functions, such as 8-bit parallel addition and subtraction, decimal adjustment, ORing, ANDing, exclusive ORing, and processing functions, such as bit shift, bit rotate, digit shift. The carry flag (C) and the zero flag (Z) change according to the results of those processings. The conditional branching command determines where to go according to the flags C and Z.

I/O

KO0 to KO15 ..... Key strobe output ports

These are ports for output only and are mainly used for key strobe, when "1" is written to a bit of the key output buffer (FOH and F1H on the internal memory), a "high" level signal is outputted through the port that corresponds to that bit. When "0" is written, a "low" level signal is outputted.

The "low" level output resistance of the KO port is quite high, i.e. 100 kΩ typ. (5 kΩ max. for the "high" level output). The reason is to prevent the ground and the VGG from short-circuited in case the key is depressed more than once. The port can be used as a normal port by lowering the "low" level output registance down to 5 kΩ max. by PLA setting change.

In addition, when "1" is set to the KSD (bit 2 of FEH on the internal memory) of the LCD contrast control, a "low" level signal is outputted with the output resistance of 5 kΩ max. in disregard of what data the key output buffer contains. Whether or not the individual bits are effective for this function is selectable by the PLA.

KI0 to KI7 .... Key input ports

These are ports for input only and are mainly used for key reading. When a "high" level signal is inputted to a KI port, "1" is set to the corresponding bit of the key input buffer (F2H on the internal memory). When a "low" level signal is inputted, "0" is set.

These ports can each be set by the PLA for whether or not to have a pull-down resistor of 125 kΩ typ. for each bit.

In addition, by inputting a "high" level signal to a KI port, "1" is set to the interrupt status register KEYI bit (bit 2 on FCH of the internal memory) so that the CPU is released from the HALT or OFF state if it is in that state. The individual KI ports can each be set by the PLA for whether or not the port is provided with the function.

ON .... ON key input port

This is a port for input only and is mainly used for key reading. While the port is at the "high" level, the ONK (bit 3 of F2H on the internal memory) of the system status register keeps "1". While it is at the "low" level, the ONK keeps "0".

The port is provided with a pull-down resistor of 125 kΩ typ.

When a "high" level signal is inputted to the ON port, the ONKI (bit 3 of FCH on the internal memory) of the interrupt status register becomes "1" so that the CPU is released from the HALT/OFF state if it is in that state.

IRQ .... Interrupt request

This pin receives an interrupt request from the outside. When "1" (active-high/active-low pull-up/pull-down is selectable by the PLA) is inputted to this pin. The interrupt control register's ISE bit (bit 6 of FCH on the internal memory) becomes "1".

When the system control register ISE bit (bit 7 of FDH on the internal memory) is "1", the CPU is released from the HALT or OFF state if it is in that state.

E0 to E15 .... General-purpose I/O ports

These general-purpose I/O ports may each be used as an input-only or output-only port by changing the PLA setting.

When "1" is written to a bit of the E port output buffer (F3H and F4H on the internal memory), a "high" level signal is outputted from the port which corresponds to that bit. When "0" is written, a "low" level signal is outputted. When a "high" level signal is inputted to an E port, the bit of the E port input buffer (F5H and F6H on the internal memory) corresponding to that port becomes "1". When a "low" level signal is inputted, it becomes "0".

If the E port is used as an input-only port, the corresponding bit of the E port output buffer is a don't-care bit. Whether or not to provide a pull-down resistor is selectable by the PLA.

If the E port is used as an output-only port, it can be used as an P-ch/N-ch open drain output port by changing the PLA setting.

If the E port is used as an I/O port, so set the PLA as to make the "low" level output resistance be high, i.e., 125 kΩ typ. (1 kΩ max. for the "high" level output) so that a "low" level signal output pulls down the input port to allow an input to the port though a "high" level signal output does not allow an input to the port (i.e. a "high" level signal must not be outputted while an output is made by the other side). Note therefore that in that case, the "low" level output resistance is so high as to decrease the speed of response.

CI/CO .... CMT I/O port

The CI port is a pin for CMT input only and has a built-in zero-cross-detector. Writing "1" to the BZ2 (bit 6 of FDH on the internal memory) of the system control register makes the CI port ready for receive an input. The status of the pin is shown by the system status register CI bit (bit 2 of FFH on the internal memory). Since the zero-cross-detector consumes the current of several-hundred µA when it is in action, set the CI port so that it becomes ready for receiving an input only when necessary. The CI port can be used as an ordinary input port by changing the PLA setting.

The CO port is a port for CMT output only, and is controlled by the BZ0 to BZ2 bits (bits 4 to 6 of FDH on the internal memory) of the system control register. The output is selectable among 2 kHz, 4 kHz, low, high, and CI. The output buffer is an ordinary C-MOS output buffer.

RXD/TXD .... UART I/O pin

The RXD is a pin for input to the UART only, and the TXD is a pin for output to the UART only.

The UART control register (F7H on the internal memory) controls the UART status, and the UART status register (F8H on the internal memory) checks the UART status. Data inputted to the UART through the RXD pin appears in the UART receive buffer, and data written in the UART transmit buffer is outputted through the TXD pin.

For each of these pins, whether active-high or active-low is selectable by changing the PLA setting. For the RXD pin, pull-up/pull-down is selectable by changing the PLA setting.

## Control Signals

### External memory control signals

MRQ .... Memory request

An external memory access request signal. Normally active-low.

RD ..... Memory read

An external memory read request signal. Normally active-low.

WR ..... Memory write

An external memory write request signal. Normally active-low.

### SH-26/LH-5073A1 control signals

ØA ..... Address latch clock

A clock used for causing the LCD driver to latch the signal on the DIO as an address. Normally active-low.

HA ..... Display clock

This provides the LCD driver with a display clock.

DIS .... Display on/off control and synchronization

The LCD driver is in "display ON mode" when the DIS is at "high" level, and in "display OFF mode" when at "low" level. The signal level of this pin is controlled by the value which the system control register DISC bit (bit 0 of FDH on the internal memory) has. When the signal at the DIS pin goes "high" from the "low" level, the CPU common output signal and the CLD driver segment output signal are both synchronized. For details, see "System Control Register" under "Internal Memory Space".

## ICE Control Signals

When the external memory is accessed, the following data is placed on the DIO by the 04 clock (See Timing Chart) for the ICE:

| | | |
|---|---|---|
| IC | D7 | Indicates the state where an interrupt starts. |
| OP | D6 | Indicates the OP-code fetch state. |
| WR | D5 | Indicates the write state |
| A19 to<br>A15 | D4 to<br>D0 | Address signal |

**Details for the ICE control signals are omitted.**

System Control

HALT .... System halt

Executing the HALT command causes the CPU to stop the main clock so that the system is brought to a halt though the sub-clock continues operating. For the data retained during the halt of the system, see the table provided below. The system is released from the halt state when any one of the following events occurs:

1. "High" level signal input to the ON pin.
2. "High" level signal input to any of the pins K0 to K7 (pin selection is allowed by the PLA).
3. The interrupt status register ST1 bit (bit 1 of FCH on the internal memory) becomes "1" when the sub-CG time expires.
4. Logical "1" is inputted to the IRQ terminal (Selectable by the PLA) while the system control register ISE bit (bit 7 of FDH on the internal memory) is "1".
5. "High" level signal input to the RESET pin.

If the system is released by any one of the events 1 to 4 above, the CPU restarts execution from the command written next to the HALT command. The time required from the event occurrence to the execution restart is several milliseconds (depending on the PLA). If the system is released by the event 5, the CPU resets the system.

If any of the events 1 to 4 has already occurred prior to an execution of the HALT command, the system is not halted but the CPU performs the NOP action by 2 states and resumes execution of the next written commands.

OFF .... System stop

Executing the OFF command causes the CPU to stop both the main clock and the sub-clock so that the system stops operating. For the data retained while the system keeps stopped, see the table provided below. The system is released from the OFF state when any one of the following events occurs:

1. "High" level signal input to the ON terminal.
2. "High" level signal input to any of the pins K0 to K7 (Pin selection is allowed by the PLA).
3. Logical "1" is inputted to the IRQ pin while the system control register ISE bit (bit 7 of FDH on the internal memory) is "1" (Selectable by the PLA).
4. "High" level signal input to the RESET pin.

If the system is released from the OFF state by any one of the events 1 to 3, the CPU restarts execution from the command written next to the OFF command. The time required from the occurrence of the event to the command execution restart is several milliseconds (depending on the PLA). If it is released by the event 4, the CPU resets the system.

If any of the events 1 to 3 has already occurred when the OFF command is executed or if the interrupt status register STI bit (bit 1 of FCH of the internal memory) is already "1", the system does not get into the OFF state but the CPU performs the NOP action by 2 stages and resumes execution of the commands written next to the OFF command.

RESET .... System initialization and restart

Inputting a "high" level signal to the RESET pin initializes the system and then restarts it. For the data initialized/retained by the system initialization, see the table below. The system is automatically restarted after executing the RESET command. The RESET command fetches the reset vector (FFFFDH to FFFFFH on the external memory) and transfers it to the program counter PC. The system is resettable even when in the HALT/OFF state.

TEST .... Test mode

When a "high" level signal is inputted to the TEST pin, the CPU is put in hardware test mode. There are the following modes available according to the statuses of the RESET pin and ON pin:

| TEST | RESET | ON | Mode | Description |
|---|---|---|---|---|
| 0 | 0 | X | Normal mode | Normal execution mode. |
| 0 | 1 | X | Reset mode | Resets the system. |
| 1 | 0 | 0 | Timer check mode | Shortens the timer setting. |
| 1 | 0 | 1 | Timer sync. mode | Synchronizes the clock. |
| 1 | 1 | X | F-ROM dump mode | Output the currently executed data in the F-ROM to the A0 to A16 and CE0 to CE4 (Partly changed by the PLE). |

The test modes above are all for hardware checking. The TEST pin requires to be normally connected to the VGG.

Detailed descriptions on the individual test modes are omitted.

System Initialization and Data Retainment for HALT/OFF/RESET

| | HALT | OFF | RESET |
|---|---|---|---|
| Registers | All retained. | All retained. | The PC read the reset vector. Others than the PC are all retained. |
| Flag (C/Z) | Undefined. | Undefined. | Retained. |
| Internal memory | USR (F8H) bits 0 to 2/5 are reset (to "0").<br>SSR (FFH) bit 2 and USR (F8H) bits 3 and 4 are set (to "1").<br>Others than the above are all retained. | USR (F8H) bits 0 to 2/5 are reset (to "0").<br>SSR (FFH) bit 2 and USR (F8H) bits 3 and 4 are set (to "1").<br>Others than the above are all retained. | ACM (FEH) bit 7, UCR (F7H), USR (F8H) bits 0 to 2/5, IMR (FCH), SCR (FDH), and SSR (FFH) bit 2 are all reset (to "0"). USR (F8H) bits 3 and 4 are set (to "1").<br>Others than the above are all retained. |

## UART

The UART converts paraller data into serial data and sends the converted data through the TDX pin. It also receives serial data from the RDX pin and converts the received data into parallel data.

The UART is controlled through the UART control register (F7H of the internal memory). The function that can be set are as follows:

| | |
|---|---|
| BAUD RATE | 300/600/1200/2400/4800/9600/19200 bps |
| PARITY | EVEN/ODD/NO |
| CHARACTER LEHGTH | 8/7 |
| STOP BITS | 1/2 |
| BREAK OUT | ON/OFF |

(For details of the setting procedures, see "LCR" under "Internal Memory".)

The UART status is check through the UART status register (F8H of the internal memory.

| | |
|---|---|
| Receive buffer status | Receiver ready |
| Transmit buffer status | Transmitter empty/Transmitter ready |
| Receive data error check | Framing error/Overrun error/ Parity error |

(For detailed checking items, see "USR" under "Internal Memory".)

The receive buffer means the UART transmit buffer whose address is F9H on the internal memory, and the transmit buffer means the UART transmit buffer whose address is FAH on the internal memory.

The UART starts tc cperate when the baud rate factors BR0 to 2 are set to other than "000". Setting the BR0 to 2 to "000" puts the UART out of operation and resets the inside of the UART.

The UART clock originates from the main CG and does not therefore works while the system is in HALT/OFF state.

When "1" is set to the Receiver Ready bit and the Transmitter Ready bit, the Receiver Ready Interrupt or Transmitter Ready Interrupt bit of the interrupt status register (whose address is FCH on the internal memory) becomes 1 so that an interrupt is requested (depending on the PLA).

Interrupt

The ESR-L has the following interrupt factors:

| | |
|---|---|
| External interrupt | Interrupt requested from the IR0 pin |
| Receiver ready interrupt | Interrupt caused by completion of 1 character receiving by the UART. |
| Transmitter ready interrupt | Interrupt caused by completion of 1 character transmission by the UART. |
| ON key interrupt | Interrupt caused by inputting a "high" level signal to the ON pin. |
| Key interrupt | Interrupt caused by inputting a "high" level signal to the KI pin. |
| SEC timer interrupt | Interrupt requested by the sub-CG timer. |
| MSEC timer interrupt | Interrupt requested by the main CG timer. |

If any one of the factors above causes an interrupt request, the bit of the interrupt status register (FCH on the internal memory) that corresponds to that interrupt factor becomes "1". If, at the same time, the bit of the interrupt mask register (FBH on the internal memory) that corresponds to the said bit and the MSB are both "1", the interrupt requested occurs.

The interrupt procedures are such that (1) The 5-byte data in the program counter PCL, PCH, PS, flag F (C: bit 0 Z: bit 1 0: bits 2 to 7), and interrupt mask register is pushed to the system stack in the order as mentioned, (2) The interrupt mask register IRM bit (bit 7) is set to "1", and (3) the data of the interrupt vector (FFFFAH to FFFFFH on the external memory) is read into the program counter.

Control return from the interrupt routine is achieved by the RETI command. The RETI command POPS the 5-byte data previously pushed to the system stacks and returns the data to the original registers.

While executing commands, the CPU reeds the operation code of the command to be executed next in one of the states where commands are executed (for most cases, in the state where the last command is executed) and judges whether to execute the command or to operate an interrupt. If there is an interrupt request already exisiting, the CPU performs that interrupt after the currently executed command.

The CPU allows an interrupt even directly after executing the RETI command. Therefore, if there are more than one interrupt request in succession, main routine execution may completely stops.

Note: If a command to change the data in the interrupt mask register is executed, an interrupt occurring directly after the execution of that command is judged according to the interrupt mask information which the register had till the command execution. Accordingly, if a command is executed to mask a certain interrupt factor, an interrupt by that factor may occur immediately after the execution of the command. In such a case, which factor has caused the interrupt cannot be determined in the interrupt routine. Hence, it is necessary program the interrupt routine to let itself return without doing anything if there are not any factors whose interrupt status register bit and interrupt mask register bit are both "1".

## Liquid Crystal Display

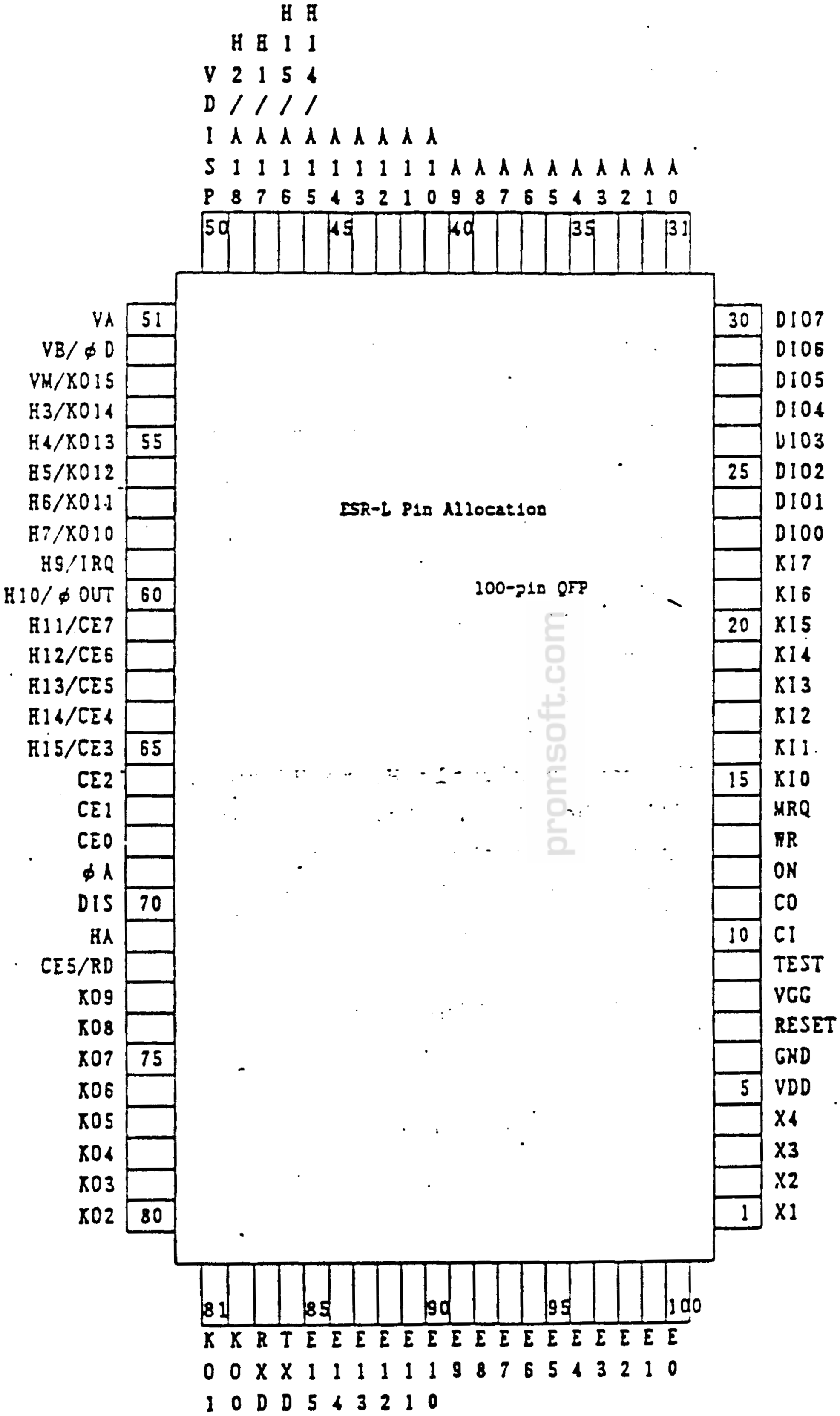
The liquid crystal display is provided with a built-in bleeder resistor that generates LCD supply voltage and a built-in display contrast regulating volume so that it is capable of generating 1/16-duty, 1/5-bias LCD supply voltage (VA, VM, VB, and VDISP) and display common signals (H1 to H7 and H9 to H15 .... Note that not all the 16 pins are used for output).

Therefore, the liquid crystal display system is constructible by simply connecting the SH-26 or LH-5073A1. (The SH-26 is HITACHI make and the CH-5073A1 is an LCD segment driver of TENRI make). The SH-26 and the LH-5073A1 are controlled by the HA, DIS, and ØA signals. For details, see "Control Signals".

The liquid crystal display contrast is controlled by the LCD contrast control register. For details, see "LCC" under "Internal Memory".

Notes: The common signals H1 to H7 and H9 to H15 makes a PLA with other I/O ports, etc. For that reason, peripheral system construction is subjected to various restrictions if the common signals are used. For details, see "Pin List".

In addition, think of using HITACHI's general-purpose LCD driver HD-61202/3 depending on peripheral system construction conditions or for a model that has a large display screen.


ESR-L Pin Allocation
100-pin QFP
VDISP 50
H2/A18
H1/A17
H15/A16
H14/A15
A14 45
A13
A12
A11
A10
A9 40
A8
A7
A6
A5
A4 35
A3
A2
A1
A0 31
VA 51
VB/ϕD
VM/KO15
H3/KO14
H4/KO13 55
H5/KO12
H6/KO11
H7/KO10
H9/IRQ
H10/ϕOUT 60
H11/CE7
H12/CE6
H13/CE5
H14/CE4
H15/CE3 65
CE2
CE1
CE0
ϕA
DIS 70
HA
CE5/RD
KO9
KO8
KO7 75
KO6
KO5
KO4
KO3
KO2 80
30 DIO7
DIO6
DIO5
DIO4
DIO3
25 DIO2
DIO1
DIO0
KI7
KI6
20 KI5
KI4
KI3
KI2
KI1
15 KI0
MRQ
WR
ON
CO
10 CI
TEST
VGG
RESET
GND
5 VDD
X4
X3
X2
1 X1
81 KO01
KO00
RXD
TXD
85 E15
E14
E13
E12
E11
90 E10
E9
E8
E7
E6
95 E5
E4
E3
E2
E1
100 E0

Pin List

| Nos. | Name | Function | When halted | When off | When reset |
|---|---|---|---|---|---|
| 1 | X1 | Main clock oscillation output | VGG | VGG | Oscillation |
| 2 | X2 | Main clock oscillation input | | | |
| 3 | X3 | Sub-clock oscillation output | Oscillation | GND/oscillation | Oscillation |
| 4 | X4 | Sub-clock oscillation input | | | |
| 5 | VDD | Display power | VGG | GND | VGG |
| 6 | GND | System power | | | |
| 7 | RESET | System reset input | | | |
| 8 | VGG | System power (-5.0 V) | | | |
| 9 | TEST | Test input | | | |
| 10 | CI | CMT input | | | |
| 11 | CO | CMT output | Software control | Software control | |
| 12 | ON | ON input | | | |
| 13 | WR | Memory control write output | Non-active | Non-active | Non-active |
| 14 | MRQ | Memory request output | Non-active | Non-active | Non-active |
| 15 to 22 | KI0 to KI7 | Key input | | | |
| 23 to 30 | DIO0 to DIO7 | Data bus input/output | Pull-up | Pull-up | Pull-up |
| 31 to 49 | A0 to A18 | Address bus output | VGG | VGG | VGG |
| 46 to 49,<br>54 to 65 | H1 to H7,<br>H9 to H15 | LCD common signal output | Common signal | GND | Common signal |
| 50 | VDISP | Display power | VDISP | GND | VDISP |
| 51 to 53 | VA, VM, VB | Display power | VA, VM, VB, waves | GND | VA, VM, VB waves |
| 52 | øD | Sub-clock waveform output | Oscillation | GND/VGG/oscillation | Oscillation |
| 53 to 56,<br>73 to 82 | KO0 to KO15 | Key output | Status retainment | Status retainment | Status retainment |

| os. | Name | Function | When halted | When off | When reset |
|---|---|---|---|---|---|
|  | IRQ | Interrupt request input |  |  |  |
|  | øOUT | Main clock waveform output | VGG | VGG | Oscillation |
| to<br>, | CE0 to CE7 | Chip enable output | Non-active | Non-active | Non-active |
|  | øA | Address latch clock output | Non-active | Non-active | Non-active |
|  | DIS | LCD driver control signal output | Software control | Software control | VGG |
|  | HA | Display synchronizing signal output | Synchronizing signal | Undefined | Synchronizing signal |
|  | RD | Memory control read output | Non-active | Non-active | Non-active |
|  | RXD | UART input |  |  |  |
|  | TXD | UART output | Software control | Software control | Pull-up/pull-down |
| to<br>0 | E0 to E15 | General input/output | Status retainment | Status retainment | Status retainment |

## ESR-L Command Settings and Operations

The explanations are given below first on the addressing modes of ESR-L and later on the cperation of each command. Refer to the separate "ESR-L Command Table" for details of assembler codes and execution state numbers in each command.

### Addressing Mode

A. Immediate value

```
mv  a,34h
```

Data itself is expressed by immediate value. Depending on the length at the destination, 1 to 3 bytes length is available.

B. Internal memory

1. Direct

```
mv  a,(5ah)
```

Address in the internal memory space is specified directly. As the internal memory has 256 bytes area, the address is designated by the unit of 1 byte.

2. BP indexed

```
mv  a,(bp+5ah)
```

The value, which is obtained by adding or subtracting 1 byte immediate value to or from BP pointer content, specifies the address in the internal memory space.

3. PX/PY indexed

```
mv  a,(px+23h)
mv  (bp),(py+5ah)
```

The value, which is obtained by adding or subtracting 1 byte immediate value to or from PX/PY pointer content, specifies the address in the internal memory space.

4. BP indexed with PX/PY offset

mv a,(bp+px)
mv (23h),(bp+py)

The value, which is obtained by adding PX or PY content to BP pointer, specifies the address in the internal memory space.

Caution) PX pointer is used only when the pointer is directed to the second operand under the internal memory addressing command in both first and second operands. In all other cases, PX pointer should be used.

C. External memory

1. Direct

mv a,[89ab3h]

Address in the external memory space is specified directly. As the external memory has 1 Mbyte storage, the address is designated by the unit of 3 bytes. In this case, the first 4 bits are "DON'T CARE".

2. Register indexed

mv a,[x]

Address in the external memory is specified by the register content. Automatical addition or subtraction of the register can also be made.

mv a, [x++]

After data transmission, the register is incremented.

mv a, [--x]

Data is transmitted after the register is decremented.

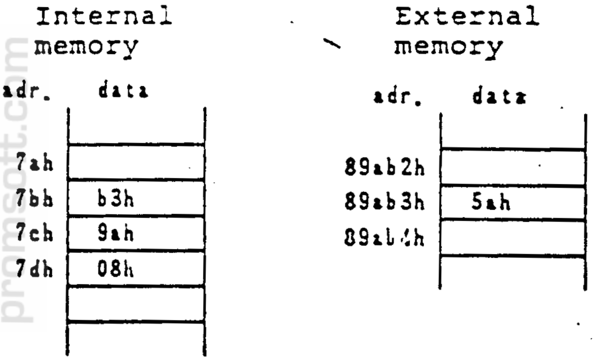
When the destination of data is either 2 bytes or 3 bytes, addition or subtraction is made for the same length of the data.

3. Internal memory indexed

mV a,[(7bh)]

The address in the external memory is specified by the successive 3 bytes data starting from the specified address in the internal memory. In this case, the smaller-number address in the internal memory denotes the low-order address in the external memory, while the larger-number address in the internal memory denotes the high-order address in the external memory. The first 4 bits are "DON'T CARE".

In the above example of 'mv a, [(7bh)],



Internal memory

| adr. | data |
|---|---|
| | |
| 7ah | |
| 7bh | b3h |
| 7ch | 9ah |
| 7dh | 08h |
| | |
| | |

External memory

| adr. | data |
|---|---|
| | |
| 89ab2h | |
| 89ab3h | 5ah |
| 89ab4h | |
| | |

The address in the external memory is specified by the data 089ab3h contained in 7bh - 7dh in the internal memory. The data of that address thus obtained, 5ah, is transmitted to accumulator.

4. Register indexed with offset

mv a,[x+23h]

Address in the external memory is specified by the value obtained by adding 1 byte data expressed in immediate value to the register content.

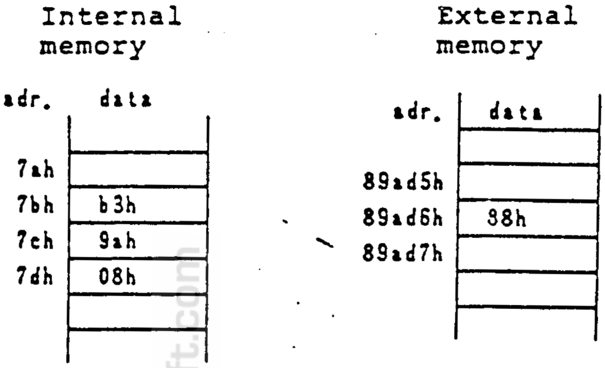
The offset value range is ±255.

5. Internal memory indexed with offset

mv a,[(7bh)+23h] Address in the external memory is specified by the value obtained by adding 1 byte data expressed in immediate value to the successive 3 bytes data of the specified address in the internal memory.

The offset value range is ±255.

In the example of 'mv a, [(7bh)] +23h',



The address in the external memory, 089ad6h, is specified by adding offset value 23h to the data 089ab3h contained in 7bh - 7dh in the internal memory. The content of the address thus specified, 88h, is transmitted to accumulator.

## 1 Byte Data Transmission Command

1. mv register,source Data transmission to register

   1 byte data is transmitted from the source to register A or register IL. When the data is transmitted to register A, the content of register B is unchanged. When the data is transmitted to register IL, the first 8 bits of register I become "0".

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | o | o | |

   Note) 'mv a,il' or 'mv il,a' is prohibited.

2. mv (*),source Data transmission in internal memory

   1 byte data is transmitted from the source to the internal memory.
   Any addressing mode of the internal memory is enabled for (*).

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | o | o | A IL |

3. mv [*],source Data transmission to external memory

   1 byte data is transmitted from the source to the external memory.
   Any addressing mode of the external memory is enabled for [*].

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | | o | | A IL |

4. mv b,a
   mv a,b Data transmission between register A and register B

   1 byte data is transmitted from register A to register B, or vice versa.

Bytes Data Transmission Command

mv  register,source    Data transmission to register

2 bytes data is transmitted from the source to register BA or register I.

When the data (including immediate value) is transmitted from the memory to the register, the smaller-number address data in the memory is sent to the low-order address of the register while the larger-number address data in the memory to the high-order address of the register.

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | o | o | BA I |

Supplement) Like 'mv  ba,x', 20 bits register can be used as the source.
In this case, only 16 bits data in the low-order address of the source register are transmitted to the destination register, with the first 4 bits data being neglected.

mvw  (*),source    Data transmission to internal
mv   (*),source    memory

2 bytes data is transmitted from the source to the internal memory. The destination is the address specified by (*) and the adjacent address of larger number.

Any adressing mode of the internal memory is enabled for (*).

When the data in transmitted from the register to the memory, low-order address data of the register is sent to the smaller-number address in the memory while high-order address data of the register to the larger-number address in the memory.

When the data is transmitted inside of the memory, the smaller-number address data of the source is sent to the smaller-number address of the destination, and the larger-number address data of the source to the larger-number address of the destination.

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | o | o | BA I |

Note) When the data source is memory (including immediate value), mnemonic is 'mvw' to indicate 2 bytes data transmission.
When the source is register, mnemonic is 'mv'.

3. mvw [*],source Data transmission to external
mv [*],source memory

2 bytes data is transmitted from the source to the external memory. The destination is the address specified by [*] and the adjacent address of larger number.

Any addressing mode of the external memory is enabled for [*].

When the data is transmitted from the register to the memory, low-order address data of the register is sent to the smaller-number address in the memory while high-order address data of the register to the larger-number address in the memory.

When the data is transmitted inside of the memory, the smaller-number address data of the source is sent to the smaller-number address in the destination, and the larger-number address data of the source to the larger-number address in the memory.

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | | o | | BA I |

Note) When the data source is memory (including immediate value), mnemonic is 'mvw' to indicate 2 bytes data transmission. When the source is register, mnemonic is 'mv'.

3 Bytes Data Transmission Command

1. mv register,source    Data transmission to register
   3 bytes data is transmitted from the source to any of the registers X, Y, U or S.
   When the data is transmitted from the memory to the register (including immediate value), the smaller-number address data in the memory is sent to the low-order address of the register, while the larger-number address data in the memory to the high-order address of the register.

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | o | o *1 | X Y U S |

*1) When the destination is register S, only direct addressing is enabled for the external memory addressing.

Supplement 1) Like 'mv x,ba', 16 bits register can be used as the source. In this case, the first 4 bits of the destination register become "0".

Supplement 2) Like 'mv x,x', the data can be transmitted to the register which is used to specify the address.

2. mvp (*),source    Data transmission to internal
   mv (*),source    memory

   3 bytes data is transmitted from the source to the internal memory. The destination is the address specified by (*) and the two adjacent addresses of larger number. Any addressing mode of the internal memory is enabled for (*). When the data is transmitted from the register to the memory, low-order address data of the register is sent to the smaller-number address in the memory, and high-order address data of the register to the larger-number address in the memory. When the data is transmitted inside of the memory, the smaller-number

address data of the source is sent to the smaller-number address of the destination, and the larger-number address data to the larger-number address.

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | o | o | X Y U S |

Note) When the data source is memory (including immediate value), mnemonic is 'mvp' to indicate 3 bytes data transmission. When the source is register, mnemonic is 'mv'.

3. mvp [*],source  Data transmission to external
   mv  [*],source  memory

3 bytes data is transmitted from the source to the external memory. The destination is the address specified by [*] and the two adjacent addresses of larger number. Any addressing mode of the external memory is enabled for [*]. When the data is transmitted from the register to the memory, low-order address data of the register is sent to the smaller-number address in the memory, and high-order address data to the larger-number address. When the data is transmitted inside of the memory, the smaller-number address data of the source is sent to the smaller-number address of the destination, and the larger-number address data to the larger-number address.

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | | | o | X Y U S *2 |

Note) When the data source is memory (including immediate value), mnemonic is 'mvp' to indicate 3 bytes data transmission. When the source is register, mnemonic is 'mv'.

*2) When the source is register S, only direct addressing is enabled for the addressing mode of the destination.

## Block Data Transmission Command

When the block data transmission command is executed, the data corresponding to the number of bytes contained in register I is transmitted from memory to memory. From the source to the destination, the data is transmitted starting from the specified address and then by incrementing the address on each side. When every 1 byte is transmitted, register I address is decremented until it becomes "0". When addressing mode is [--reg.] at either source or destination, special process is followed which should be noted.

Note 1) After completion of block transmission command, the content of register I becomes "0". When the block transmission command is executed with register I being "0", 65536 bytes data is transmitted.

Note 2) While one command is being executed in ESR-L, no interrupt operation can be performed. Therefore, due attention should be paid to interruption interval when transmitting voluminous data by giving block transmission command.

Note 3) The block transmission command, which transmits data inside of the internal memory, includes 'mvl (*),(*)' and 'mvld (*),(*)'. Either of these needs to be selected depending on the direction of data transmission. (The details will be described below.)

1. mvl (*),source  Data transmission to internal memory

The data corresponding to the number of bytes contained in register I is transmitted from the source to the internal memory. Any addressing mode of the internal memory is enabled for (*).

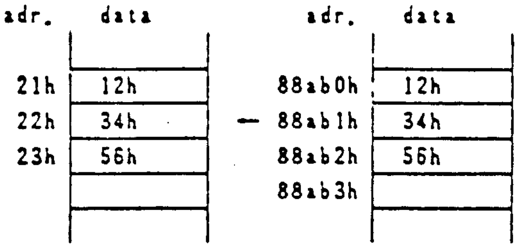
Only when the addressing mode of the source is [--reg.], the transmission is repeated until register I becomes "0" by decrementing the source side address starting from smaller side adjacent address of the specified one, and at the same time by incrementing the destination side address starting from the specified one.

In case of the operation like;

```
  mv    x,88ab3h
  mv    il,3
  mvl   (23h),[--x]
```

the data transmission will be done as shown in the right.

After completion, x will be 38ab0h.



Note that the destination side is not pre-decrement.

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | | o | o *1 | |

*1) OPE-CODO which corresponds to 'mvl (*),[reg.]' does not exist. In this case use 'mvl (*),[reg.+0].

2. mvl [*],source    Data transmission to external memory

The data corresponding to the number of bytes contained in register I is transmitted from the source to the external memory. Any addressing mode of the internal memory is enabled for [*].

Only when the addressing mode of the destination is [--reg.], the transmission is repeated until register I becomes "0", by incrementing the source side address starting from the specified one, and at the same time by decrementing the destination side address starting from smaller side adjacent address of the specified one.

| Source | Immediate value | Internal memorv | External memory | Registe: |
|---|---|---|---|---|
| | | o | | |

*2) OPE-CODE which corresponds to 'mvl [reg.],source' does not exist. In this case 'mvl [reg.+0],source'.

3. mvld (*),(*) Data transmission inside of internal memory

In 'mvl (*),(*)', the data is transmitted while incrementing the address at both source and destination sides.

As an example, take the case that 3 bytes data out of 5 bytes need to be transmitted to the larger-number address. After the first data has been transmitted, the fourth byte data which is not yet transmitted will be destroyed. To avoid this, therefore, the data should be transmitted by starting from the larger-number address and then decrementing one by one. In 'mvld (*),(*)', the data is transmitted by starting from the specified address in both source and destination, and then decrementing the address. The number of bytes to be transmitted is the whcle content of the register I. Any addressing mode of the internal memory is enabled for (*).

Exchange Command

1. ex a,b Data exchange between register A and register B

The contents of register A and register B are exchanged with each other.

2. ex register,register Data exchange between registers

Data is exchanged between the registers. Basically, the first operand and the second operand should be both 16 bits register or both 20 bits register. When the registers cf different capacity are used as source and destination, the ¢ases like following will occur. When 'mv ba,x' command is executed, low-order side 16 bits data of register X is transmitted to register BA, with the first 4 bits data being missed. At the same time, 16 bits data of register BA is transmitted to register X, with the first 4 bits of register X becoming "0".

3. ex (*),(*) Data exchange between internal memories (1 byte)

exw (*),(*) Data exchange between internal memories (2 bytes)

exp (*),(*) Data exchange between internal memories (3 bytes)

exl (*),(*) Data exchange between internal memories (n byte: Block exchange)

Data is exchanged between the internal memories. Depending on the number of data to be exchanged, mnemonic should be used separately as shown above. When the data of 2 bytes or more is exchanged, the data in the address larger than the specified address is exchanged for the specified number of bytes in both first and second operands. Attention should be paid to avoid overlapping of the data specified by the first operand and the data specified by the second operand. Any addressing mode of the internal memory is enabled for (*).

Note 1) After completion of the block exchange command, the content of register I becomes "0". When the block exchange command is executed with register I being "0", 65536 (64K) bytes data is exchanged.

Note 2) While one command is being executed in ESR-L, no interrupt operation can be performed. Therefore, due attention should be paid to interruption interval when exchanging voluminous data by giving exchange command.

## Dyadic Operation No.1 (Add/Subtract Command)

When the add/subtract command is executed, the source data is added to or subtracted from the destination data, and the result is stored in the destination. Addition and subtraction have both binary operation and decimal operation.

When the carry occurs as a result of operation, C (Carry) FLAG becomes "1". Otherwise it becomes "0". When the operation result is "0", Z (Zero) FLAG becomes "1". Otherwise it becomes "0". When the consecutive operation command (adcl, sbcl, dadl, dsbl) is executed, Z FLAG becomes "1" only when all digits of the operation result are "0". Otherwise Z FLAG becomes "0".

1. add a,source — Addition to accumulator
   sub a,source — Subtraction from accumulator

   The source data is added to or subtracted from accumulator data.

| Source | Immediate value | Internal memory | External memorv | Register |
|---|---|---|---|---|
| | o | o | | A IL |

2. adc a,source — Addition to accumulator
   sbc a,source — Subtraction from accumulator

   The source data and C FLAG data are added to or subtracted from accumulator data.

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | o | | |

3. add register,register — Addition between registers
   sub register,register — Subtraction between registers

   The source register data is added to or subtracted from the destination register data. Though any of the registers A, IL, BA, I, X, Y, U and S can be used, the destination register should be same as or larger than the source register. When the destination register is larger than the source register, the operation is performed by regarding the lacked high-order bits of the source register as "0".

C FLAG is determined by the carry from the top bit of the destination register. In case of X, Y, U and S registers, therefore, it depends on the carry from the 20th bit. Z FLAG becomes "1" when every bit of the destination register is made "0" as a result of the operation. Otherwise Z FLAG becomes "0".

4. add (*),source — Addition to internal memory
sub (*),source — Subtraction from internal memory

The source data is added to or subtracted from the internal memory data.
Any addressing mode of the internal memory is enabled for (*).

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | ○ | | | A |

5. adc (*),source — Addition to internal memory
sbc (*),source — Subtraction from internal memory

The source data and C FLAG data are added to or subtracted from the internal memory data.
Any addressing mode of the internal memory is enabled for (*).

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | ○ | | | A |

6. adcl (*),(*) — Consecutive addition between internal memories
sbcl (*),(*) — Consecutive subtraction between internal memories
dadl (*),(*) — Consecutive addition between internal memories (Decimal)
dsbl (*),(*) — Consecutive subtraction between internal memories (Decimal)

The source internal memory data is consecutively added to or subtracted from the destination internal memory data. Starting from the address specified by (*), the operation is made by incrementing the address in case of binary operation and by decrementing the address in case of decimal operation at each of destination and source. In every 1 byte operation, register I is decremented until it becomes "0". At the first byte, only the destination data and the source data are added or subtracted. From the second byte onward, C FLAG data which is the result of former operation is also added or subtracted together.
Any addressing mode of the internal memory is enabled for (*).

7. adcl (*),a — Consecutive addition of register A to internal memory

sbcl (*),a — Consecutive subtraction of register A from internal memory

dadl (*),a — Consecutive addition of register A to internal memory (Decimal)

dsbl (*),a — Consecutive subtraction of register A from internal memory (Decimal)

The register A data is consecutively added to or subtracted from the internal memory data. The register A data is added to or subtracted from the data in the address specified by (*). "0" is added to or subtracted from the internal memory data while incrementing the address in case of binary operation and decrementing the address in case of decimal operation. In every one byte operation, register I is decremented until it becomes "0". At the first byte, the register A data is added to or subtracted from the internal memory data. From the second byte onward, only C FLAG which is the result of former operation is added or subtracted.
Any addressing mode of the internal memory is enabled for (*).

Dyadic Operation No.2 (Logical Operation Command)

When the logical operation command is executed, destination data and source data are put to logical operation and the result is stored in the destination.

When the operation result is "0", Z (Zero) FLAG becomes "1". Otherwise it becomes "0". C (Carry) FLAG content is kept unchanged.

```
1.  and  a,source           AND to accumulator
    or   a,source           OR to accumulator
    xor  a,source           Exclusive OR to accumulator
```

The accumulator data and the source data are put to logical operation.

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | o | | |

```
2.  and  (*),source         AND to internal memory
    or   (*),source         OR to internal memory
    xor  (*),source         Exclusive OR to internal memory
```

The internal memory data and the source data are put to logical operation.

Any addressing mode of the internal memory is enabled for (*).

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | o | | A |

```
3.  and  [*],source         AND to external memory
    or   [*],source         OR to external memory
    xor  [*],source         Exclusive OR to external memory
```

The external memory data and the source data are put to logical operation.

Among the addressing modes of the external memory, only direct addressing is enabled for [*].

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | | | |

Dyadic Operation No.3 (Pointer Add Command)

Data is added to the internal memory without changing C/Z FLAGS.

The command is used primarily to make relative change cf the contents of pointers BP, PX, PY, etc.

pmdf (*),source        Addition to internal memory
The source data is added to the internal memory data.
During the operation, C/Z FLAGS are kept unchanged.
Any addressing mode of the internal memory is enabled for (*).

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
|  | o |  |  | A |

Dyadic Operation No.4 (Compare Command)

When the compare command is executed, the source data is subtracted from the destination data, and according to its result C/Z (Carry/Zero) FLAGS are changed. When the carry occurs as a result of the operation, C FLAG becomes "1". Otherwise it becomes "0". When the operation result is "0", Z FLAG becomes "1". Otherwise it becomes "0". When the consecutive operation command (cmpw, cmpp) is executed, Z FLAG becomes "1" only when all operation results are "0", i.e., when all digits of the operation result are "0". Otherwise it becomes "0".

When more than 2 bytes data in the internal memory are compared, the data in those addresses smaller in address number than the specified one is put to comparison. The address smaller in address number is the low-order address, and the address larger in address number is the high-order address. The operation result is not stored anywhere.

1. cmp a,source     Comparison with accumulator

   The accumulator data and the source data are compared.

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | | | |

2. cmp (*),source     Comparison with 1 byte data of internal memory

   1 byte data of internal memory and the source data are compared.

   Any addressing mode of the internal memory is enabled for (*).

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | o | | A |

3. cmpw (*),source Comparison with 2 bytes data of internal memory

2 bytes data of internal memory and the source data are compared.

Any addressing mode of the internal memory is enabled for (*).

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | | o | | BA I |

4. cmpp (*),source Comparison with 3 bytes data of internal memory

3 bytes data of internal memory and the source data are compared.

Any addressing mode of the internal memory is enabled for (*).

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | | o | | XYUS |

Note) When the source is a 20 bits register as in case of 'cmpp (23h),x', for example, the comparison is made for 24 bits as well. The empty high-order 4 bits of the register are regarded as "0" in the comparison.

5. cmp [*],source Comparison with external memory

The external memory data and the source data are compared.

Among the addressing modes of the external memory, only direct addressing is enabled for [*].

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | | | |

Dyadic Operation No.5 (Bit Test Command)

When the bit test command is executed, destination data and source data are ANDed.. Depending on the operation result, Z (Zero) FLAG is changed. When the operation result is "0", Z FLAG becomes "1". Otherwise it becomes "0". C (Carry) FLAG is kept unchanged.
The operation result is not stored anywhere.

1. test a,source Bit test of accumulator
Accumulator data is subjected to the bit test with source data.

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | | | |

2. test (*),source Bit test of internal memory
Internal memory data is subjected to the bit test with source data.
Any addressing mode of the internal memory is enabled for (*).

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | | | A |

3. test [*],source Bit test of external memory
External memory data is subjected to the bit test with source data.
Among the addressing modes of the external memory, only direct addressing is enabled for [*].

| Source | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | o | | | |

Monadic Operation

1. swap a                    Digit exchange of accumulator

High-order 4 bits and low-order 4 bits of the accumulator data are exchanged with each other.

When the accumulator data is "0", Z (Zero) FLAG becomes "1". Otherwise it becomes "0".

C (Carry) FLAG is kept unchanged.

2. inc operand          Increment
dec operand          Decrement

Operand is incremented or decremented by "1".

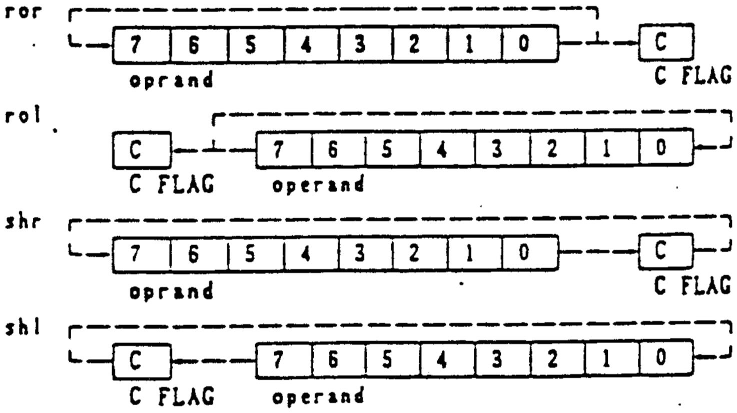
When the result is "0", Z FLAG becomes "1". Otherwise it becomes "0".

C FLAG is kept unchanged.

| Operand | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | | o | | A IL BA I X Y U S |

3. ror operand          Rightward bit rotation
rol operand          Leftward bit rotation
shr operand          Rightward bit shift
shl operand          Leftward bit shift

Operand data is given the bit movement as shown below.



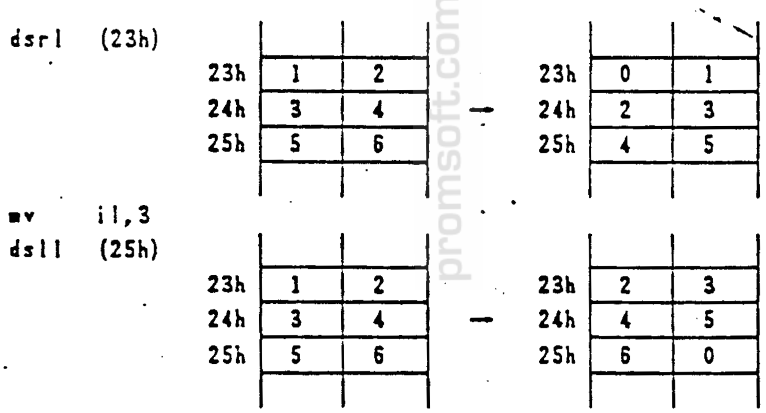
When the operand is "0" as a result of bit movement, Z FLAG becomes "1". Otherwise it becomes "0".

| Operand | Immediate value | Internal memory | External memory | Register |
|---|---|---|---|---|
| | | o | | A |

4. dsrl (*) Rightward digit shift cf internal memory
   dsll (*) Leftward digit shift of internal memory

The internal memory data is given the digit shift as shown below. The shift is made by the number of bytes specified by register I.
When all digits are "0" as a result of the digit shift, Z (Zero) FLAG becomes "1". Otherwise it becomes "0". C (Carry) FLAG is kept unchanged.
Any addressing mode of the internal memory is enabled for (*).

Example) mv il,3



```
dsrl  (23h)
          23h | 1 | 2 |        23h | 0 | 1 |
          24h | 3 | 4 |   —    24h | 2 | 3 |
          25h | 5 | 6 |        25h | 4 | 5 |

mv    il,3
dsll  (25h)
          23h | 1 | 2 |        23h | 2 | 3 |
          24h | 3 | 4 |   —    24h | 4 | 5 |
          25h | 5 | 6 |        25h | 6 | 0 |
```

"0" is given to the digit which is emptied by the shift. The digit which is shifted last is cleared.
Due attention should be paid to the fact that 'dsrl' increments the address, starting from the specified one and 'dsll' decrements the address in the same manner as above to repeat the shift.

C FLAG Control Command

sc Setting of carry flag

rc Resetting of carry flag
'sc' sets C FLAG to "1", while 'rc' sets it to "0". Z FLAG is kept unchanged.

## PUSH POP Command

ESR-L has two stacks. One is S (System) stack which is primarily used to store the necessary data while CPU is making subroutine call or interrupt. Another is U (User) stack which is used freely by the user.

1. pushu operand — Push command to U stack
   popu operand — Pop command from U stack

   'pushu' transmits the operand data to the smaller-number address adjacent to the one specified by register U. When the operand is a 16/20 bits register, the data is transmitted by decrementing the address, starting from the smaller-number address adjacent the one specified by register U. After the transmission, the content of register U has been decremented by the number of bytes transmitted.

   'popu' transmits the data in the address specified by register U to the operand. When the operand is a 16/20 bits register, the data is transmitted by incrementing the address, starting from the address specified by register U. After the transmission, the content of register U has been incremented by the number of bytes transmitted.

   Like the 'mv' command, the high-order byte of the register corresponds to the larger-number address in the memory and the low-order byte to the smaller-number address.

   The following can be operands.

   f FLAG : C (bit 0), Z (bit 1), 0 (bits 2 to 7)

   imr (0fbh) : Interrupt mask register (Internal memory FBH)

   a/il/ba/i/x/y : Register

   Note) When 'popu il' is executed, the high-order 8 bits of register I become "0".

2. pushs operand Push command to S stack<br>
pops operand Pop command from S stack

The functions of these commands are quite the same as 'pushu' and 'popu'. However, the address is specified by register S instead of register U.

What can be the cperands for the commands are also the same as those for 'pushu' and 'popu'. In terms cf hardware, however, no OPE-CODE exists which enables cperands to correspond to imr/register. Instead, the assembler enables the 'mv' command which performs the same functicn.

```
pushs  imr                 mv   [--s],(0fbh)
pops   imr          -->    mv    (0fbh),[s++]
pushs  register            mv   [--s],register
pops   register            mv   register,[s++]
```

JUMP Command

1.

| | | | |
|---|---|---|---|
| jr | ±n | Unconditional relative branch | n: Immediate value of 1 byte |
| jrz | ±n | Conditional relative branch (Z="1") | |
| jrnz | ±n | Conditional relative branch (Z="0") | |
| jrc | ±n | Conditional relative branch (C="1") | |
| jrnc | ±n | Conditional relative branch (C="0") | |

The relative branch command changes, the content of program counter (PC) to a value obtained by adding 2 to the address where the command's mnemonic is written and then adding ±n to the sum. This means that since n is an immediate value of 1 byte, branching can be implemented within the address range from -253 bytes to +257 bytes counted from the address where the branch command is written.

Conditional relative branching is implemented when a corresponding condition is satisfied. When the condition is not met, the command becomes NOP.

Note) Address counting is made only in the PC, and no carry is made to the page segment register (PS). This means that the relative branch command cannot cause control to jump out of the segment.

2.

| | | | |
|---|---|---|---|
| jp | mn | Unconditional branch inside of segment | mn: Immediate value of 2 bytes |
| jpz | mn | Conditional branch inside of segment (Z="1") | |
| jpnz | mn | Conditional branch inside of segment (Z="0") | |
| jpc | mn | Conditional branch inside of segment (C="1") | |
| jpnc | mn | Conditional branch inside of segment (C="0") | |

Branching is implemented by reading a 2 bytes immediate value into PC. At this time, the larger-number address data of the immediate value is read into the high-order side of PC, and the smaller-number address data into the low-order side.

Conditional branching is implemented when a corresponding condition is satisfied. When the condition is not met, the command becomes NOP.

3. jpf operand Branch outside of segment
Branching is implemented to the whole 1 Mbyte memory area by reading 3 bytes of data from the operand into PC and PS. When the source is memory, the larger-number address data in the memory is read into the high-order side of PC, and the smaller-number address data into the low-order side.

| Source | Immediate value | Internal memory | External memory | Registe |
|---|---|---|---|---|
| | o | o | | X Y U S |

CALL RET Command

1. call mn — Subroutine call inside of segment
mn: Immediate value of 2 bytes

The address of the command next to this command (the address which is 3 bytes larger than the address where the 'call' command is written) is pushed into S stack, and a 2 bytes immediate value is read into PC (program counter). At this time, the larger-number address data of the immediate value is read into the high-order side of PC, and the smaller-number address data into the low-order side.

The low-order 16 bits (2 bytes) are pushed out of the 20 bits address.

2. callf lmn — Subroutine call outside of segment
lmn: Immediate value of 3 bytes

The address of the command next to this command (the address which is 4 bytes larger than the address where the 'callf' command is written) is pushed into S stack, and a 3 bytes immediate value is read into PC and PS (pace segment register). At this time, the largest-number address data of the immediate value is read into PS. Out of the remaining 2 bytes data, the larger-number address data is read into the high-order side of PC, and the smaller-number address data into the low-order side.

The 20 bits address is pushed by 3 bytes. At this time, the highest-order 4 bits are "0".

3. ret — Return against call

2 bytes data is popped from S stack into PC.

4. retf — Return against callf

3 bytes data is popped from S stack into PC and PS.

Supplement) Refer to "PUSH POP Command" for the details of PUSH and POP.

RESET INTERRUPT Command

1. reset Reset

   3 bytes data in addresses FFFFDH to FFFFFH is read into PC (program counter) and PS (page segment register). (The data in address FFFFFH is read into PS; and the data in FFFFEH is read into the high-order side of PC and the data in FFFFDH into the low-order side of PC.)

2. int Interrupt

   Following the order of PS, PC, FLAG nad IMR (FBH of internal memory), their contents (a total of 5 bytes) are pushed into S stack. 3 bytes data in addresses FFFFAH to FFFFCH is read into PC and PS. (The data in address FFFFCH is read into PS; and the data in FFFFBH is read into the high-order side of PC, and the data in FFFFAH into the low-order side of PC.)

   After IMR is pushed, IRM (bit 7) of IMR becomes "0".

3. reti Return interrupt

   5 bytes data is popped from S stack and then put into IMR, FLAG, PC and PS in this order.

Supplement) Refer to "PUSH POP Command" for the details of PUSH and POP.

CPU Control Command

1. wait                Wait

    Decrement is repeated until register I becomes "0". Except for decrementing register I, the command is NOP.

    Note) While one command is being executed in ESR-L, any interrupt operation is not performed. Due attention should be paid to interruption interval when the operation is suspended for a long period of time by using the wait command.

2. nop                NOP

    No operation is performed during one state.

3. tcl                Timer clear

    When this command is executed by setting "1" to STCL (bit 1) and MTCL (bit 0) of LCC (FEH in internal memory), SUB CG timer and MAIN CG timer are reset respectively.

4. halt                CPU halt

    CPU is paused by halting MAIN CG of CPU. Refer to 'System Control HALT' for the details of the HALT condition.

5. off                CPU off

    CPU is turned off by stopping MAIN CG and SUB CG of CPU. Refer to 'System Control OFF' for the details of the OFF condition.

# ESR-L Command Table

## Reference Guide

| | |
|---|---|
| 1 | 2 |
| 3 | |
| 4 | |

1. MNEMONIC

2. FLAG CHANGE ... CARRY/ZERO
   C,Z: Changes according to the state with the execution of a command.
   1,0: Becomes 1 or 0.
   . Does not change.

3. CODE .......... OP-CODE and POST-BYTE are expressed by binary form, and other operands are expressed by l, m, n, L, M and N (byte). Among OP-CODE and POST-BYTE, -r- and -R- are changed depending on the register as shown below.

| -r- | -R- | CODE |
|---|---|---|
| a | A | 000 |
| il | IL | 001 |
| ba | BA | 010 |
| i | I | 011 |
| x | X | 100 |
| y | Y | 101 |
| u | U | 110 |
| s | S | 111 |

4. NUMBER OF EXECUTION STATES ... As to addressing for internal memory, the number shown is for the case when (bp+n) is used.
   When any other addressing mode is used, 1 state should be added.

## 1 Byte Transmission Command Nc.1

32 PRE-BYTE

[illegible] ⇒ [illegible]

| Source<br>Destination | n | a<br>il | (n)<br>(bp±n)<br>(px±n)<br>(py±n)<br>(bp+px)<br>(bp+py) | [lmn] | [x]<br>[y]<br>[u]<br>[s] |
|---|---|---|---|---|---|
| A<br>IL | mv ./.<br>00001-R-,n<br>a:2/il:3 | | mv ./.<br>10000-R-,n<br>a:3/il:4 | mv ./.<br>10001-R-,nml<br>6 | mv ./.<br>10010-R-,<br>00000-r-<br>a:4/il:5 |
| (N)<br>(bp+N)<br>(px+N)<br>(bp+px) | mv ./.<br>11001100,Nn<br>3 | mv ./.<br>10100-r-,N<br>3 | mv ./.<br>11001000,Nn<br>6 | mv ./.<br>11010000,<br>Nnml<br>7 | mv ./.<br>11100000,<br>00000-r-,N<br>6 |
| [LMN] | | mv ./.<br>10101-r-,NML<br>5 | mv ./.<br>11011000,<br>NMLn<br>6 | | |
| [X]<br>[Y]<br>[U]<br>[S] | | mv ./.<br>10110-r-,<br>00000-R-<br>4 | mv ./.<br>11101000,<br>00000-R-,n<br>6 | | |
| [X++]<br>[Y++]<br>[U++]<br>[S++] | | mv ./.<br>10110-r-,<br>00100-R-<br>4 | mv ./.<br>11101000,<br>00100-R-,n<br>6 | | |
| [−−X]<br>[−−Y]<br>[−−U]<br>[−−S] | | mv ./.<br>10110-r-,<br>00110-R-<br>5 | mv ./.<br>11101000,<br>00110-R-,n<br>7 | | |
| [X±N]<br>[Y±N]<br>[U±N]<br>[S±N] | | mv ./.<br>10110-r-,<br>1s000-R-,N<br>6 | mv ./.<br>11101000,<br>1s000-R-,nN<br>8 | | |
| [(N)]<br>[(bp±N)]<br>[(px±N)]<br>[(bp+px)] | | mv ./.<br>10111-r-,<br>00000000,N<br>9 | mv ./.<br>11111000,<br>00000000,Nn<br>11 | | |
| [(M)±N]<br>[(bp±M)±N]<br>[(px±M)±N]<br>[(bp+px)±N] | | mv ./.<br>10111-r-,<br>1s000000,MN<br>11 | mv ./.<br>11111000,<br>1s000000,MnN<br>13 | | |

1 Byte Transmission Command No.2

| Source<br>Destination | [x++]<br>[y++]<br>[u++]<br>[s++] | [−−x]<br>[−−y]<br>[−−u]<br>[−−s] | [x±n]<br>[y±n]<br>[u±n]<br>[s±n] | [(n)]<br>[(bp±n)]<br>[(px±n)]<br>[(py±n)]<br>[(bp+px)]<br>[(bp+py)] | [(m)±n]<br>[(bp±m)±n]<br>[(px±m)±n]<br>[(py±m)±n]<br>[(bp+px)±n]<br>[(bp+py)±n] |
|---|---|---|---|---|---|
| A<br>IL | mv ./.<br>10010-R-,<br>00100-r-<br>a:4/il:5 | mv ./.<br>10010-R-,<br>00110-r-<br>a:5/il:6 | mv ./.<br>10010-R-,<br>1s000-r-,n<br>a:6/il:7 | mv ./.<br>10011-R-,<br>00000000,n<br>a:9/il:10 | mv ./.<br>10011-R-,<br>1s000000,mn<br>a:11/il:12 |
| (N)<br>(bp+N)<br>(px+N)<br>(bp+px) | mv ./.<br>11100000,<br>00100-r-,N<br>6 | mv ./.<br>11100000,<br>00110-r-,N<br>7 | mv ./.<br>11100000,<br>1s000-r-,Nn<br>8 | mv ./.<br>11110000,<br>00000000,Nn<br>11 | mv ./.<br>11110000,<br>1s000000,Nmn<br>13 |
| [LMN] | | | | | |
| [X]<br>[Y]<br>[U]<br>[S] | | | | | |
| [X++]<br>[Y++]<br>[U++]<br>[S++] | | | | | |
| [−−X]<br>[−−Y]<br>[−−U]<br>[−−S] | | | | | |
| [X±N]<br>[Y±N]<br>[U±N]<br>[S±N] | | | | | |
| [(N)]<br>[(bp±N)]<br>[(px±N)]<br>[(bp+px)] | | | | | |
| [(M)±N]<br>[(bp±M)±N]<br>[(px±M)±N]<br>[(bp+px)±N] | | | | | |

2 Byte Transmission Command No.1

| Source<br>Destination | mn | ba<br>i | (n)<br>(bp±n)<br>(px±n)<br>(py±n)<br>(bp+px)<br>(bp+py) | [lmn] | [x]<br>[y]<br>[u]<br>[s] |
|---|---|---|---|---|---|
| BA<br>I | mv ./.<br>00001-R-,nm<br>3 | mv ./.<br>11111101,<br>0-R-0-r-<br>2 | mv ./.<br>10000-R-,n<br>4 | mv ./.<br>10001-R-,nml<br>7 | mv ./.<br>10010-R-,<br>00000-r-<br>5 |
| (N)<br>(bp±N)<br>(px±N)<br>(bp+px) | mvw ./.<br>11001101,Nnm<br>4 | mv ./.<br>10100-r-,N<br>4 | mvw ./.<br>11001001,Nn<br>8 | mvw ./.<br>11010001,<br>Nnml<br>8 | mvw ./.<br>11100001,<br>00000-r-,N<br>7 |
| [LMN] |  | mv ./.<br>10101-r-,NML<br>6 | mvw ./.<br>11011001,<br>NMLn<br>7 |  |  |
| [X]<br>[Y]<br>[U]<br>[S] |  | mv ./.<br>10110-r-,<br>00000-R-<br>5 | mvw ./.<br>11101001,<br>00000-R-,n<br>7 |  |  |
| [X++]<br>[Y++]<br>[U++]<br>[S++] |  | mv ./.<br>10110-r-,<br>00100-R-<br>5 | mvw ./.<br>11101001,<br>00100-R-,n<br>7 |  |  |
| [--X]<br>[--Y]<br>[--U]<br>[--S] |  | mv ./.<br>10110-r-,<br>00110-R-<br>6 | mvw ./.<br>11101001,<br>00110-R-,n<br>10 |  |  |
| [X±N]<br>[Y±N]<br>[U±N]<br>[S±N] |  | mv ./.<br>10110-r-,<br>1s000-R-,N<br>7 | mvw ./.<br>11101001,<br>1s000-R-,nN<br>9 |  |  |
| [(N)]<br>[(bp±N)]<br>[(px±N)]<br>[(bp+px)] |  | mv ./.<br>10111-r-,<br>00000000,N<br>10 | mvw ./.<br>11111001,<br>00000000,Nn<br>12 |  |  |
| [(M)±N]<br>[(bp±M)±N]<br>[(px±M)±N]<br>[(bp+px)±N] |  | mv ./.<br>10111-r-,<br>1s000000,MN<br>12 | mvw ./.<br>11111001,<br>1s000000,MnN<br>14 |  |  |

2 Byte Transmission Command No.2

| Source<br>Destination | [x++]<br>[y++]<br>[u++]<br>[s++] | [--x]<br>[--y]<br>[--u]<br>[--s] | [x±n]<br>[y±n]<br>[u±n]<br>[s±n] | [(n)]<br>[(bp±n)]<br>[(px±n)]<br>[(py±n)]<br>[(bp+px)]<br>[(bp+py)] | [(m)±n]<br>[(bp±m)±n]<br>[(px±m)±n]<br>[(py±m)±n]<br>[(bp+px)±n]<br>[(bp+py)±n] |
|---|---|---|---|---|---|
| BA<br>I | mv ./.<br>10010-R-,<br>00100-r-<br>5 | mv ./.<br>10010-R-,<br>00110-r-<br>6 | mv ./.<br>10010-R-,<br>1s000-r-,n<br>7 | mv ./.<br>10011-R-,<br>00000000,n<br>10 | mv ./.<br>10011-R-,<br>1s000000,mn<br>12 |
| (N)<br>(bp+N)<br>(px+N)<br>(bp+px) | mvw ./.<br>11100001,<br>00100-r-,N<br>7 | mvw ./.<br>11100001,<br>00110-r-,N<br>10 | mvw ./.<br>11100001,<br>1s000-r-,Nn<br>9 | mvw ./.<br>11110001,<br>00000000,Nn<br>12 | mvw ./.<br>11110001,<br>1s000000,Nmn<br>14 |
| [LMN] | | | | | |
| [X]<br>[Y]<br>[U]<br>[S] | | | | | |
| [X++]<br>[Y++]<br>[U++]<br>[S++] | | | | | |
| [--X]<br>[--Y]<br>[--U]<br>[--S] | | | | | |
| [X±N]<br>[Y±N]<br>[U±N]<br>[S±N] | | | | | |
| [(N)]<br>[(bp±N)]<br>[(px±N)]<br>[(bp+px)] | | | | | |
| [(M)±N]<br>[(bp±M)±N]<br>[(px±M)±N]<br>[(bp+px)±N] | | | | | |

3 Byte Transmission Command No.1

| Source<br>Destination | lmn | x<br>y<br>u<br>s | (n)<br>(bp±n)<br>(px±n)<br>(py±n)<br>(bp+px)<br>(bp+py) | [lmn] | [x]<br>[y]<br>[u]<br>[s] |
|---|---|---|---|---|---|
| X<br>Y<br>U<br>S | mv ./.<br>00001-R-,nml<br>4 | mv ./.<br>11111101,<br>0-R-0-r-<br>2 | mv ./.<br>10000-R-,n<br>5 | mv ./.<br>10001-R-,nml<br>8 | mv ./.<br>10010-R-,<br>00000-r-<br>6 |
| (N)<br>(bp+N)<br>(px+N)<br>(bp+px) | mvp ./.<br>11011100,<br>Nnml<br>5 | mv ./.<br>10100-r-,N<br>5 | mvp ./.<br>11001010,Nn<br>10 | mvp ./.<br>11010010,<br>Nnml<br>9 | mvp ./.<br>11100010,<br>00000-r-,N<br>8 |
| [LMN] |  | mv ./.<br>10101-r-,NML<br>7 | mvp ./.<br>11011010,<br>NMLn<br>8 |  |  |
| [X]<br>[Y]<br>[U]<br>[S] |  | mv ./.<br>10110-r-,<br>00000-R-<br>6 | mvp ./.<br>11101010,<br>00000-R-,n<br>8 |  |  |
| [X++]<br>[Y++]<br>[U++]<br>[S++] |  | mv ./.<br>10110-r-,<br>00100-R-<br>7 | mvp ./.<br>11101010,<br>00100-R-,n<br>9 |  |  |
| [--X]<br>[--Y]<br>[--U]<br>[--S] |  | mv ./.<br>10110-r-,<br>00110-R-<br>8 | mvp ./.<br>11101010,<br>00110-R-,n<br>10 |  |  |
| [X±N]<br>[Y±N]<br>[U±N]<br>[S±N] |  | mv ./.<br>10110-r-,<br>1s000-R-,N<br>8 | mvp ./.<br>11101010,<br>1s000-R-,nN<br>10 |  |  |
| [(N)]<br>[(bp±N)]<br>[(px±N)]<br>[(bp+px)] |  | mv ./.<br>10111-r-,<br>00000000,N<br>11 | mvp ./.<br>11111010,<br>00000000,Nn<br>13 |  |  |
| [(M)±N]<br>[(bp±M)±N]<br>[(px±M)±N]<br>[(bp+px)±N] |  | mv ./.<br>10111-r-,<br>1s000000,MN<br>13 | mvp ./.<br>11111010,<br>1s000000,MnN<br>15 |  |  |

s is excluded for -r-.
S is excluded for -R-.

3 Byte Transmission Command No.2

| Source<br>Destination | [x++]<br>[y++]<br>[u++]<br>[s++] | [−−x]<br>[−−y]<br>[−−u]<br>[−−s] | [x±n]<br>[y±n]<br>[u±n]<br>[s±n] | [(n)]<br>[(bp±n)]<br>[(px±n)]<br>[(py±n)]<br>[(bp+px)]<br>[(bp+py)] | [(m)±n]<br>[(bp±m)±n]<br>[(px±m)±n]<br>[(py±m)±n]<br>[(bp+px)±n]<br>[(bp+py)±n] |
|---|---|---|---|---|---|
| X<br>Y<br>U<br>S | mv ./.<br>10010-R-,<br>00100-r-<br>7 | mv ./.<br>10010-R-,<br>00110-r-<br>8 | mv ./.<br>10010-R-,<br>1s000-r-, n<br>8 | mv ./.<br>10011-R-,<br>00000000, n<br>11 | mv ./.<br>10011-R-,<br>1s000000, mn<br>13 |
| (N)<br>(bp+N)<br>(px+N)<br>(bp+px) | mvp ./.<br>11100010,<br>00100-r-, N<br>9 | mvp ./.<br>11100010,<br>00110-r-, N<br>10 | mvp ./.<br>11100010,<br>1s000-r-, Nn<br>10 | mvp ./.<br>11110010,<br>00000000, Nn<br>13 | mvp ./.<br>11110010,<br>1s000000, Nmn<br>15 |
| [LMN] | | | | | |
| [X]<br>[Y]<br>[U]<br>[S] | | | | | |
| [X++]<br>[Y++]<br>[U++]<br>[S++] | | | | | |
| [−−X]<br>[−−Y]<br>[−−U]<br>[−−S] | | | | | |
| [X±N]<br>[Y±N]<br>[U±N]<br>[S±N] | | | | | |
| [(N)]<br>[(bp±N)]<br>[(px±N)]<br>[(bp+px)] | | | | | |
| [(M)±N]<br>[(bp±M)±N]<br>[(px±M)±N]<br>[(bp+px)±N] | | | | | |

S is excluded for -R-.

Block Transmission Command

| Source<br>Destination | (n)<br>(bp±n)<br>(px±n)<br>(py±n)<br>(bp+px)<br>(bp+py) | [lmn] | [x++]<br>[y++]<br>[u++]<br>[s++] | [−−x]<br>[−−y]<br>[−−u]<br>[−−s] | [x±n]<br>[y±n]<br>[u±n]<br>[s±n] |
|---|---|---|---|---|---|
| (N)<br>(bp÷N)<br>(px÷N)<br>(bp÷px) | mvl/mvld ./.<br>11001011,Nn/<br>11001111,Nn<br>5+2×i | mvl ./.<br>11010011,<br>Nnml<br>6+2×i | mvl ./.<br>11100011,<br>00100-r-,N<br>5+2×i | mvl ./.<br>11100011,<br>00110-r-,N<br>7÷2×i | mvl ./.<br>01010110,<br>1s000-r-,Nn<br>5÷2×i |
| [LMN] | mvl ./.<br>11011011,NMLn<br>6+2×i | | | | |
| [X++]<br>[Y++]<br>[U++]<br>[S++] | mvl ./.<br>11101011,<br>00100-R-,n<br>5+2×i | | | | |
| [−−X]<br>[−−Y]<br>[−−U]<br>[−−S] | mvl ./.<br>11101011,<br>00110-R-,n<br>7+2×i | | | | |
| [X±N]<br>[Y±N]<br>[U±N]<br>[S±N] | mvl ./.<br>01011110,<br>1s000-R-,nN<br>5+2×i | | | | |
| [(N)]<br>[(bp±N)]<br>[(px±N)]<br>[(bp+px)] | mvl ./.<br>11111011,<br>00000000,Nn<br>10+2×i | | | | |
| [(M)±N]<br>[(bp±M)±N]<br>[(px±M)±N]<br>[(bp+px)±N] | mvl ./.<br>11111011,<br>1s000000,MnN<br>12+2×i | | | | |

| source<br>Destination | [(n)]<br>[(bp±n)]<br>[(px±n)]<br>[(py±n)]<br>[(bp+px)]<br>[(bp+py)] | [(m)±n]<br>[(bp±m)±n]<br>[(px±m)±n]<br>[(py±m)±n)<br>[(bp+px)±n)<br>[(bp+py)±n) |
|---|---|---|
| (N)<br>(bp+N)<br>(px+N)<br>(bp+px) | mvl ./.<br>11110011,<br>00000000,Nn<br>10+2×i | mvl ./.<br>11110011,<br>1s000000,Nmn<br>12÷2×i |
| | | |

Transmission Command Between a and b

| Source<br>Destination | a | b |
|---|---|---|
| A |  | mv ./.<br>01110100<br>1 |
| B | mv ./.<br>01110101<br>1 |  |

Exchange Command

|  | a<br>b | ba<br>i | x<br>y<br>u<br>s | (n)<br>(bp±n)<br>(px±n)<br>(py±n)<br>(bp+px)<br>(bp+py) |  |
|---|---|---|---|---|---|
| A<br>B | ex ./.<br>11011101<br>3 |  |  |  |  |
| BA<br>I |  | ex ./.<br>11101101,<br>0-R-0-r-<br>4 |  |  |  |
| X<br>Y<br>U<br>S |  |  | ex ./.<br>11101101,<br>0-R-0-r-<br>4 |  |  |
| (N)<br>(bp±N)<br>(px±N)<br>(bp+px) |  |  |  | ex ./.<br>11000000,Nn<br>7 | exw ./.<br>11000001,Nn<br>10 |
|  |  |  |  | exp ./.<br>11000010,Nn<br>13 | exl ./.<br>11000011,Nn<br>5+3×i |

Dyadic Cperation No.1 (Add/Subtract Command)

| Source<br>Destination | n | a | il | ba<br>i | x<br>y<br>u<br>s | (n)<br>(bp±n)<br>(px±n)<br>(py±n)<br>(bp+px)<br>(bp+py) |
|---|---|---|---|---|---|---|
| A | add/sub c/z<br>01000000,n/<br>01001000,n<br>3 | add/sub c/z<br>01000110,0-R-0-r-/<br>01001110,0-R-0-r-<br>3 | | | | add/sub c/z<br>01000010,n/<br>01001010,n<br>4 |
| | adc/sbc c/z<br>01010000,n/<br>01011000,n<br>3 | | | | | adc/sbc c/z<br>01010010,n/<br>01011010,n<br>4 |
| IL | | | | | | |
| BA<br>I | | add/sub c/z<br>01000100,0-R-0-r-/<br>01001100,0-R-0-r-<br>5 | | | | |
| X<br>Y<br>U<br>S | | add/sub c/z<br>01000101,0-R-0-r-/<br>01001101,0-R-0-r-<br>7 | | | | |
| (N)<br>(bp±N)<br>(px±N)<br>(bp+px) | add/sub c/z<br>01000001,Nn/<br>01001001,Nn<br>4 | add/sub c/z<br>01000011,N/<br>01001011,N<br>4 | | | | |
| | adc/sbc c/z<br>01010001,Nn/<br>01011001,Nn<br>4 | adc/sbc c/z<br>01010011,N/<br>01011011,N<br>4 | | | | |
| | | adcl/sbcl c/z<br>01010101,N/<br>01011101,N<br>4+i | | | | adcl/sbcl c/z<br>01010100,Nn/<br>01011100,Nn<br>5+2×i |
| | | dadl/dsbl c/z<br>11000101,N/<br>11010101,N<br>4+i | | | | dadl/dsbl c/z<br>11000100,Nn/<br>11010100,Nn<br>5+2×i |

## Dyadic Operation No.2 (Logical Operation Command)

| Source<br>Destination | n | a | (n)<br>(bp± n)<br>(px± n)<br>(py± n)<br>(bp+ px)<br>(bp+ py) |
|---|---|---|---|
| A | and/or/xor ./z<br>01110000, n/01111000, n/<br>01101000, n<br>3 |  | and/or/xor ./z<br>01110111, n/01111111, n/<br>01101111, n<br>4 |
| (N)<br>(bp± N)<br>(px± N)<br>(bp+ px) | and/or/xor ./z<br>01110001, Nn/01111001,<br>Nn/01101001, Nn<br>4 | and/or/xor ./z<br>01110011, n/01111011,<br>n/01101011, n<br>4 | and/or/xor ./z<br>01110110, Nn/01111110,<br>Nn/01101110, Nn<br>6 |
| [LMN] | and/or/xor ./z<br>01110010, NMLn/01111010<br>, NMLn/01101010, NMLn<br>7 |  |  |

## Dyadic Operation No.3 (Pointer Add Command)

| Source<br>Destination | n | a |
|---|---|---|
| (N)<br>(bp± N)<br>(px± N)<br>(bp+ px) | pmdf ./.<br>01000111, Nn<br><br>4 | pmdf ./.<br>01010111, N<br><br>4 |

Dyadic Operation No.4 (Compare Command)

| Source<br>Destination | n | a | ba<br>i | x<br>y<br>u<br>s | (n)<br>(bp±n)<br>(px±n)<br>(py±n)<br>(bp+px)<br>(bp+py) |
|---|---|---|---|---|---|
| A | cmp c/z<br>01100000,n<br>3 | | | | |
| (N)<br>(bp±N)<br>(px±N)<br>(bp+px) | cmp c/z<br>01100001,Nn<br>4 | cmp c/z<br>01100011,N<br>4 | | | cmp c/z<br>10110111,Nn<br>6 |
| | | | cmpw c/z<br>11010110,<br>00000-r-,N<br>7 | | cmpw c/z<br>11000110,Nn<br>8 |
| | | | | cmpp c/z<br>11010111,<br>00000-r-,N<br>9 | cmpp c/z<br>11000111,Nn<br>10 |
| [LMN] | cmp c/z<br>01100010,<br>NMLn<br>6 | | | | |

Dyadic Operation No.5 (Test Command)

| Source<br>Destination | n | a |
|---|---|---|
| a | test ./z<br>01100100,n<br>3 | |
| (N)<br>(bp±N)<br>(px±N)<br>(bp+px) | test ./z<br>01100101,Nn<br>4 | test ./z<br>01100111,N<br>4 |
| [LMN] | test ./z<br>01100110,<br>NMLn<br>6 | |

## Monadic Operation

| Operation<br>Operand | digit<br>exchange | increment/<br>decrement | rotate | shift | digit shift |
|---|---|---|---|---|---|
| a | swap ./z<br>11101110<br>3 | inc/dec ./z<br>01101100,<br>00000-r-/<br>01111100,<br>00000-r-<br>3 | ror/rol c/z<br>11100100/<br>11100110<br>2 | shr/shl c/z<br>11110100/<br>11110110<br>2 | |
| il<br>ba i<br>x y u s | | | | | |
| (n)<br>(bp±n)<br>(px±n)<br>(bp+px) | | inc/dec ./z<br>01101101,n/<br>01111101,n<br>3 | ror/rol c/z<br>11100101,n/<br>11100111,n<br>3 | shr/shl c/z<br>11110101,n/<br>11110111,n<br>3 | dsrl/dsll ./z<br>11111100,n/<br>11101100,n<br>4+i |

## C FLAG Control Command

| set | reset |
|---|---|
| sc 1/.<br>10010111<br>1 | rc 0/.<br>10011111<br>1 |

## PUSH POP Command

| | | f | imr | a<br>il | ba<br>i | x<br>y |
|---|---|---|---|---|---|---|
| u-stack | push | pushu ./.<br>00101110<br>3 | pushu ./.<br>00101111<br>3 | pushu ./.<br>00101-r-<br>3 | pushu ./.<br>00101-r-<br>4 | pushu ./.<br>00101-r-<br>5 |
| | pop | popu c/z<br>00111110<br>2 | popu ./.<br>00111111<br>2 | popu ./.<br>00111-r-<br>a:2/il:3 | popu ./.<br>00111-r-<br>3 | popu ./.<br>00111-r-<br>4 |
| s-stack | push | pushs ./.<br>01001111<br>3 | pushs<br>mv<br>[--s],imr | pushs<br>mv [--s],r | | |
| | pop | pops c/z<br>01011111<br>2 | pops<br>mv<br>imr,[s++] | pops<br>mv r,[s++] | | |

JUMP Command

| | +n | −n | mn<br>lmn | x<br>y<br>u<br>s | (n)<br>(bp±n)<br>(px±n)<br>(bp+px) |
|---|---|---|---|---|---|
| Near<br>ever | jr ./.<br>00010010,n<br>3 | jr ./.<br>00010011,n<br>3 | jp ./.<br>00000010,nm<br>4 | | |
| zero | jrz ./.<br>00011000,n<br>j:3/nj:2 | jrz ./.<br>00011001,n<br>j:3/nj:2 | jpz ./.<br>00010100,nm<br>j:4/nj:3 | | |
| non-zero | jrnz ./.<br>00011010,n<br>j:3/nj:2 | jrnz ./.<br>00011011,n<br>j:3/nj:2 | jpnz ./.<br>00010101,nm<br>j:4/nj:3 | | |
| carry | jrc ./.<br>00011100,n<br>j:3/nj:2 | jrc ./.<br>00011101,n<br>j:3/nj:2 | jpc ./.<br>00010110,nm<br>j:4/nj:3 | | |
| non-carry | jrnc ./.<br>00011110,n<br>j:3/nj:2 | jrnc ./.<br>00011111,n<br>j:3/nj:2 | jpnc ./.<br>00010111,nm<br>j:4/nj:3 | | |
| Far | | | jpf ./.<br>00000011,nml<br>5 | jp ./.<br>00010001,<br>00000-r-<br>4 | jp ./.<br>00010000,n<br>6 |

CALL RETURN Command

| | call mn<br>lmn | return |
|---|---|---|
| Near | call ./.<br>00000100,nm<br>6 | ret ./.<br>00000110<br>4 |
| Far | callf ./.<br>00000101,nml<br>8 | retf ./.<br>00000111<br>5 |

RESET INTERRUPT Command

| reset | interrupt | return<br>interrupt |
|---|---|---|
| reset ./.<br>11111111<br>6 | int ./.<br>11111110<br>12 | reti ./.<br>00000001<br>7 |

CPU Control Command

| wait | non-<br>operation | timer<br>clear | halt | off |
|---|---|---|---|---|
| wait<br>11101111<br>1+i | nop ./.<br>00000000<br>1 | tcl ./.<br>11001110<br>1 | halt ./.<br>11011110<br>(When passing: 2) | off ./.<br>11011111<br>(When passing: 2) |

PRE-BYTE (Internal Memory Addressing Mode Setting Byte)

| 2nd operand<br>1st operand | (n) | (bp±n)<br>* | (py±n) | (bp+py) |
|---|---|---|---|---|
| (N) | — ./.<br>00110010<br>1 | — ./.<br>00110000<br>1 | — ./.<br>00110011<br>1 | — ./.<br>00110001<br>1 |
| (bp±N) | — ./.<br>00100010<br>1 | | — ./.<br>00100011<br>1 | — ./.<br>00100001<br>1 |
| (px±N) | — ./.<br>00110110<br>1 | — ./.<br>00110100<br>1 | — ./.<br>00110111<br>1 | — ./.<br>00110101<br>1 |
| (bp+px) | — ./.<br>00100110<br>1 | — ./.<br>00100100<br>1 | — ./.<br>00100111<br>1 | — ./.<br>00100101<br>1 |

*) When the number of operands for the internal memory is only one, the assembler generates PRE-BYTE shown in the column marked with * above is regardless of whether the operand is the first or second one.

# INSTRUCTION TABLE

| L \ H | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NOP | JP<br>(n) |  | PRE<br>(m)<br>(BP+n) | ADD<br>A,n | ADC<br>A,n | CMP<br>A,n | AND<br>A,n | MV<br>A,(n) | MV<br>A,[$r_3$] | MV<br>(n),A | MV<br>[$r_3$],A | EX<br>(m),(n) | MV<br>(k),[lmn] | MV<br>(m),[$r_3$] | MV<br>(m),[(n)] |
| 1 | RETI | JP<br>r3 | PRE<br>(BP+m)<br>(BP+PY) | PRE<br>(m)<br>(BP+PY) | ADD<br>(m),n | ADC<br>(m),n | CMP<br>(m),n | AND<br>(m),n | MV<br>IL,(n) | MV<br>IL,[$r_3$] | MV<br>(n),IL | MV<br>[$r_3$],IL | EXW<br>(m),(n) | MVW<br>(k),[lmn] | MVW<br>(m),[$r_3$] | MVW<br>(m),[(n)] |
| 2 | JP<br>mn | JR<br>+n | PRE<br>(BP+m)<br>(n) | PRE<br>(m)<br>(n) | ADD<br>A,(n) | ADC<br>A,(n) | CMP<br>[klm],n | AND<br>[klm],n | MV<br>BA,(n) | MV<br>BA,[$r_3$] | MV<br>(n),BA | MV<br>[$r_3$],BA | EXP<br>(m),(n) | MVP<br>(k),[lmn] | MVP<br>(m),[$r_3$] | MVP<br>(m),[(n)] |
| 3 | JPF<br>lmn | JR<br>−n | PRE<br>(BP+m)<br>(PY+n) | PRE<br>(m)<br>(PY+n) | ADD<br>(n),A | ADC<br>(n),A | CMP<br>(n),A | AND<br>(n),A | MV<br>I,(n) | MV<br>I,[$r_3$] | MV<br>(n),I | MV<br>[$r_3$],I | EXL<br>(m),(n) | MVL<br>(k),[lmn] | MVL<br>(m), [$r_3$ +<br>+] | MVL<br>(m),[(n)] |
| 4 | CALL<br>mn | JPZ<br>mn | PRE<br>(BP+PX)<br>(BP+n) | PRE<br>(PX+m)<br>(BP+n) | ADD<br>$r_2\ {}^{r_1}_{r_2}$ | ADCL<br>(m),(n) | TEST<br>A,n | MV<br>A,B | MV<br>X,(n) | MV<br>X,[$r_3$] | MV<br>(n),X | MV<br>[$r_3$],X | DADL<br>(m),(n) | DSBL<br>(m),(n) | ROR<br>A | SHR<br>A |
| 5 | CALLF<br>lmn | JPNZ<br>mn | PRE<br>(BP+PX)<br>(BP+PY) | PRE<br>(PX+m)<br>(BP+PY) | ADD<br>$r_3$,r | ADCL<br>(n),A | TEST<br>(m),n | MV<br>B,A | MV<br>Y,(n) | MV<br>Y,[$r_3$] | MV<br>(n),Y | MV<br>[$r_3$],Y | DADL<br>(n),A | DSBL<br>(n),A | ROR<br>(n) | SHR<br>(n) |
| 6 | RET | JPC<br>mn | PRE<br>(BP+PX)<br>(n) | PRE<br>(PX+m)<br>(n) | ADD<br>$r_1$,$r_1$ | MVL<br>(m), [X +<br>n] | TEST<br>[klm],n | AND<br>(m),(n) | MV<br>U,(n) | MV<br>U,[$r_3$] | MV<br>(n),U | MV<br>[$r_3$],U | CMPW<br>(m),(n) | CMPW<br>(m),$r_2$ | ROL<br>A | SHL<br>A |
| 7 | RETF | JPNC<br>mn | PRE<br>(BP+PX)<br>(PY+n) | PRE<br>(PX+m)<br>(PY+n) | PMDF<br>(m),n | PMDF<br>(m),A | TEST<br>(n),A | AND<br>A,(n) | MV<br>S,(n) | SC | MV<br>(n),S | CMP<br>(m),(n) | CMPP<br>(m),(n) | CMPP<br>(m),$r_3$ | ROL<br>(n) | SHL<br>(n) |
| 8 | MV<br>A,n | JRZ<br>+n | PUSHU<br>A | POPU<br>A | SUB<br>A,n | SBC<br>A,n | XOR<br>A,n | OR<br>A,n | MV<br>A,[lmn] | MV<br>A,[(n)] | MV<br>[lmn],A | MV<br>[(n)],A | MV<br>(m),(n) | MV<br>[lmn],(n) | MV<br>[$r_3$],(n) | MV<br>[(m)],(n) |
| 9 | MV<br>IL,n | JRZ<br>−n | PUSHU<br>IL | POPU<br>IL | SUB<br>(m),n | SBC<br>(m),n | XOR<br>(m),n | OR<br>(m),n | MV<br>IL,[lmn] | MV<br>IL,[(n)] | MV<br>[lmn],IL | MV<br>[(n)],IL | MVW<br>(m),(n) | MVW<br>[lmn],(n) | MVW<br>[$r_3$],(n) | MVW<br>[(m)],(n) |
| A | MV<br>BA,mn | JRNZ<br>+n | PUSHU<br>BA | POPU<br>BA | SUB<br>A,(n) | SBC<br>A,(n) | XOR<br>[klm],n | OR<br>[klm],n | MV<br>BA,[lmn] | MV<br>BA,[(n)] | MV<br>[lmn],BA | MV<br>[(n)],BA | MVP<br>(m),(n) | MVP<br>[lmn],(n) | MVP<br>[$r_3$],(n) | MVP<br>[(m)],(n) |
| B | MV<br>I,mn | JRNZ<br>−n | PUSHU<br>I | POPU<br>I | SUB<br>(n),A | SBC<br>(n),A | XOR<br>(n),A | OR<br>(n),A | MV<br>I,[lmn] | MV<br>I,[(n)] | MV<br>[lmn],I | MV<br>[(n)],I | MVL<br>(m),(n) | MVL<br>[lmn],(n) | MVL<br>[$r_3$ + +],<br>(n) | MVL<br>[(m)],(n) |
| C | MV<br>X,lmn | JRC<br>+n | PUSHU<br>X | POPU<br>X | SUB<br>$r_2\ {}^{r_1}_{r_2}$ | SBCL<br>(m),(n) | INC<br>r | DEC<br>r | MV<br>X,[lmn] | MV<br>X,[(n)] | MV<br>[lmn],X | MV<br>[(n)],X | MV<br>(m),n | MVP<br>(k),lmn | DSLL<br>(n) | DSRL<br>(n) |
| D | MV<br>Y,lmn | JRC<br>−n | PUSHU<br>Y | POPU<br>Y | SUB<br>$r_3$,r | SBCL<br>(n),A | INC<br>(n) | DEC<br>(n) | MV<br>Y,[lmn] | MV<br>Y,[(n)] | MV<br>[lmn],Y | MV<br>[(n)],Y | MVW<br>(l),mn | EX<br>A,B | EX<br>$r_2$,$r_2$<br>$r_3$,$r_3$ | MV<br>$r_2$,$r_2$<br>$r_3$,$r_3$ |
| E | MV<br>U,lmn | JRNC<br>+n | PUSHU<br>F | POPU<br>F | SUB<br>$r_1$,$r_1$ | MVL<br>[X + n]<br>(m) | XOR<br>(m),(n) | OR<br>(m),(n) | MV<br>U,[lmn] | MV<br>U,[(n)] | MV<br>[lmn],U | MV<br>[(n)],U | TCL | HALT | SWAP<br>A | IR |
| F | MV<br>S,lmn | JRNC<br>−n | PUSHU<br>IMR | POPU<br>IMR | PUSHS<br>F | POPS<br>F | XOR<br>A,(n) | OR<br>A,(n) | MV<br>S,[lmn] | RC | MV<br>[lmn],S |  | MVLD<br>(m),(n) | OFF | WAIT | RESET |