Epson HX-20 Software Reference Manual

EPSON Corporation

2021 remaster by José M. Tévar

Contents

Preface

This book is a complete remaster of the Epson HX-20 Software Reference Manual as can currently be found online. Source material was retrieved from the wonderful F. J. Kraan's "Oh no, not another computer museum!" website (`http://www.vintagecomputer.net/fjkraan/comp/`) where the original manual contributed by Paul Struijt has been preserved. The main problem with the original material is the fact that it consists of the scanned typewritten documents in an non-searchable PDF format. Besides, it seems the original material never went beyond the draft stage, as it is poorly laid out and was apparently not revised or proofread.

This remastered version tries to partially correct some of these issues with the original material. Specifically, the new $\LaTeX\,2_\varepsilon$-generated PDF is fully searchable and, hopefully, has a more readable layout.

One must, however, bear in mind that this is a remaster and not a rewritten or proofread version of the manual and, as such, errors and inconsistencies present in the original will most probably also be present in this book: fidelity has been the main priority, and only the most egregious typos have been corrected.

Sample code listings have been tested and correctness of the generated bytecode verified against the original's listings. The software with which said sample code has been assembled is Alfred Arnold's AS Macro Assembler (`http://john.ccac.rwth-aachen.de:8000/as/`) and the listings are in this assembler's output format. The assembly command used has been

`asw -L` [*source_filename*]

which, in a Windows system, generates an assembly listing file consisting of line numbers, source file and produced code (plua symbol table and other information not reproduced in this manual) with the same name as the source file and `.lst` extension.

Hopefully, the number of newly introduced errors in the text will be small and this book will represent a positive contribution to keeping this amazing little machine alive and kicking.

Chapter 1

General

1.1 Descriptive expressions used in this Manual

The HX-20 incorporates two HD6301 microprocessors. One of the microprocessors has a 64K-byte memory area to control the entire HX-20 components and is called the master MCU (Micro Computer Unit). The other plays an auxiliary role. Namely, it controls I/O devices such as the microprinter, cassettes, etc., and is called the slave MCU. Each MCU has a CPU, a ROM, a RAM, a serial I/O port, a parallel I/O port, and timer function.

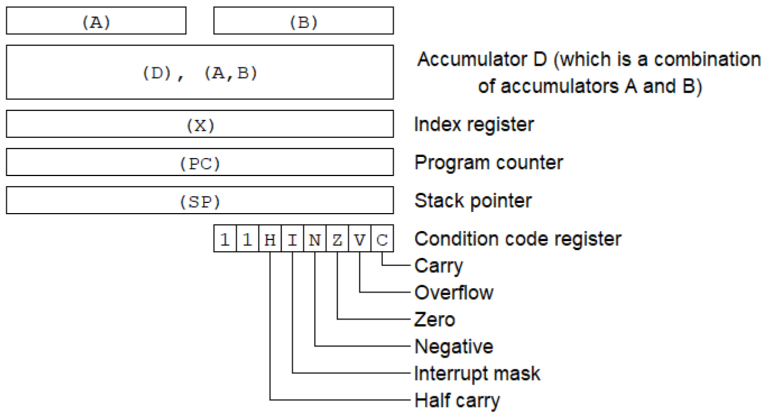


Figure 1.1: Arrangement of CPU Registers

The registers are identified by symbols: `(A)` for accumulator A, `(B)` for accumulator B, `(D)` or `(A,B)` for accumulator D, `(X)` for the index register, `(PC)` for the program counter, and `(SP)` for the stack pointer. For the condition code register, `(H)`, `(I)`, `(N)`, `(Z)`, `(V)`, and `(C)` are used to indicate the respective bits as shown in Figure 1.1. As shown in Figure 1.2, bit positions are indicated from the bit lowest place with the lowest place value (or weighting) at the extreme right such as bit 0, bit 1,... This bit with the lowest value is called LSB (Least Significant Bit), while the bit with the highest place value (at the extreme left) is called MSB (Most Significant Bit).

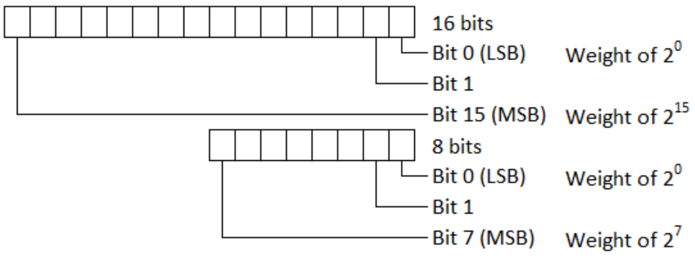
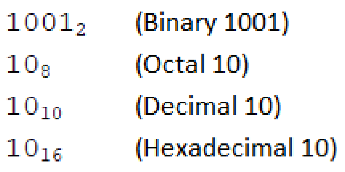


Figure 1.2: Bit Positions

As various number systems are in use, such as binary, decimal, hexadecimal, etc., the base or radix of a number system is placed as a subscript to the lower right of a number to identify whether the number is decimal, hexadecimal, octal or binary as shown in Figure 1.3.



1001_2	(Binary 1001)
10_8	(Octal 10)
10_{10}	(Decimal 10)
10_{16}	(Hexadecimal 10)

Figure 1.3: Number notation (with Base m)

Unless otherwise specified, the hexadecimal notation is used throughout this manual to express the contents of memory and registers which are represented by binary numbers. Characters represented by ASCII codes may be enclosed by single quotation marks (ex: '`ABCD`'). Symbol $\triangle$ represents a space.

The MCU is provided with the internal registers shown in Table 1.1. The registers are abbreviated as follows.

Address	Register	Abbreviation
00	Port 1 Data direction register	DDR1
01	Port 2 Data direction register	DDR2

Continues in next page...

...continued from previous page.

Address	Register	Abbreviation
02	Port 1 Data register	PORT 1
03	Port 2 Data register	PORT 2
04	Port 3 Data direction register	DDR3
05	Port 4 Data direction register	DDR4
06	Port 3 Data register	PORT 3
07	Port 4 Data register	PORT 4
08	Timer control and status register	TCSR
09	Counter (high-order byte)	FRC
0A	Counter (low-order byte)	
0B	Output compare register (high-order byte)	OCR
0C	Output compare register (low-order byte)	
0D	Input capture register (high-order byte)	ICR
0E	Input capture register (high-order byte)	
0F	Port 3 control and status register	
10	Rate mode control register	RMCR
11	Transmit/receive control and status register	TRCSR
12	Receive data register	RDR
13	Transmit data register	TDR
14	RAM control register	
15 to 1F	Reserved	

Table 1.1: Internal registers

Table 1.2 lists the abbreviations for the respective bits of each internal register.

Address register	Bit	Abbreviation
02 PORT 1	0	P10
	1	P11
	2	P12
	3	P13
	4	P14
	5	P15
	6	P16
	7	P17
03 PORT 2	0	P20
	1	P21

Continues in next page...

...continued from previous page.		
Address register	Bit	Abbreviation
06 PORT 3	0	P30
	1	P31
	2	P32
	3	P33
	4	P34
	5	P35
	6	P36
	7	P37
07 PORT 4	0	P40
	1	P41
	2	P42
	3	P43
	4	P44
	5	P45
	6	P46
	7	P47
08 TCSR	0 (Output level)	OLVL
	1 (Input edge)	IEDG
	2 (Enable timer overflow interrupt)	ETOI
	3 (Enable output compare interrupt)	EOCI
	4 (Enable input capture interrupt)	EICI
	5 (Timer overflow flag)	TOF
	6 (Output compare flag)	OCF
	7 (Input capture flag)	ICF
11 TRCSR	0 (Wake up)	WU
	1 (Transmit enable)	TE
	2 (Transmit interrupt enable)	TIE
	3 (Receive enable)	RE
	4 (Receive interrupt enable)	RIE
	5 (Transmit data register empty)	TDRE
	6 (Overrun framing error)	ORFE
	7 (Receive data register full)	

Table 1.2: Bits of internal registers and their abbreviations

1.2 Sample program format

Table 1.3 shows the standard format of a sample program.

Column number		Description
1 to 8		Source file line number (decimal)
10 to 17		Location counter value (hexadecimal)
21 to 40		Coded instruction (hexadecimal)
41 to 80		Comment line (free format)
41 to EOL	41 to 46	Label field
	47 to 52	Operation field
	54 to EOL	Operands field
		Comments (optional, after operands)

Table 1.3: Standard format of sample program

Find below a sample program, with header lines showing the standard format.

```
    5    10   15   20   25   30   35   40   45   50   55   60   65   70   75   80
----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
Line no./Location : Coded instruction   Label Opcod Operands ; Comments
      1/       0 :                           PAGE  0
      2/       0 :                           CPU   6301
      3/    1000 :                           ORG   $1000
      4/    1000 :                     ;
      5/    1000 :                     ; 16 bit unsigned multiply
      6/    1000 :                     ; (16 bit result)
      7/    1000 :                     ; reentrant code (uses 6 bytes on stack)
      8/    1000 :                     ;
      9/    1000 :                     ; A,B times X result A,B
     10/    1000 :                     ;
     11/    1000 : 37                  MPY16 PSHB
     12/    1001 : 36                        PSHA
     13/    1002 : 3C                        PSHX
     14/    1003 : 30                        TSX
     15/    1004 :                     ; Stack now looks like
     16/    1004 :                     ; +0 MS byte multiplication         A B
     17/    1004 :                     ; +1 LS byte                      * C D
     18/    1004 :                     ; +2 MS byte multiplier         --------
     19/    1004 :                     ; +3 LS byte                         BD
     20/    1004 : A6 02                     LDAA  2,X   ; A * D          AD
     21/    1006 : E6 01                     LDAB  1,X   ;                BC
     22/    1008 : 3D                        MUL         ;              AC
     23/    1009 : 37                        PSHB        ;            ---------
     24/    100A : A6 03                     LDAA  3,X   ; B * C         X Y Z
     25/    100C : E6 00                     LDAB  0,X   ;
     26/    100E : 3D                        MUL         ;
     27/    100F : 37                        PSHB        ;
     28/    1010 : A6 03                     LDAA  3,X   ; B * D
     29/    1012 : E6 01                     LDAB  1,X   ;
     30/    1014 : 3D                        MUL         ;
     31/    1015 : 30                        TSX         ;
     32/    1016 : AB 00                     ADDA  0,X
```

```
1018 : AB 01            ADDA  1,X
101A : 38               PULX        ; Clean up stack
101B : 38               PULX
101C : 38               PULX
101D : 39               RTS
101E :                  END
```

1.3 How to read subroutine lists

The subroutine lists in each chapter contain the subroutine names, entry points, descriptions of subroutines, and parameters. The parameters shown are divided into those to be output for subroutine call and those to be input for subroutine return, In describing the CPU registers, symbols are used: `(A)` for accumulator A, `(B)` for accumulator B, and `(X)` for the index register. For the condition code register, `(C)` stands for carry, `(N)` for a negative flag, and `(Z)` for a zero flag. Details for registers are listed under "Registers retained". "Subroutines referenced" lists the subroutines called in the course of execution. The I/O routines normally use addresses 0050 to 0077 as a work area. The actual locations used are represented as variables (see Chapter 14).

"`(C)`: abnormal I/O flag" appears quite often in the description of parameters at the time of subroutine return. This indicates that the I/O operation has not been performed correctly due to a drop in voltage, the power switch being turned OFF, or the BREAK key being pressed. `(C)`=1 indicates abnormal I/O operation and `(C)`=0 indicates normal I/O operation.

Chapter 2

Input from keyboard

2.1 General

The keyboard, connected to the master MCU, has 8 lines each of which fetches 10 data. In other words, the keyboard is an 8×10 matrix structure. The pressed key can be found by inputting the data for each line. Interrupt `IRQ1` occurs each time a key is pressed. The keyboard matrix incorporates the Printer ON/OFF and DIP switches in addition to the alphanumeric keys.

Key input is processed by interrupts and input data is stored in the 8-byte key stack. A power ON key stack, which stores data to be input automatically from the keyboard when the power is turned ON, is also provided. The contents of the power ON key stack are first fetched and the data in the key stack is input when the power ON key stack becomes empty. The contents of the power ON key stack can be restored by turning the power ON (reset).

2.2 Ports for keyboard input

Table 2.1 shows the I/O ports related to the keyboard input.

Address	Bit position	Definition
20	0	Output. Specifies scanning of `L0` line. `0`: Scanning enabled. `1`: Scanning is not performed.
	1	Output. `L1`
	2	Output. `L2`
	3	Output. `L3`
	4	Output. `L4`

Continues in next page...

...continued from previous page.		
Address	Bit position	Definition
	5	Output. L5
	6	Output. L6
	7	Output. L7
22	0	Input. Scan result D0. 0: ON; 1: OFF
	1	Input. Scan result D1. 0: ON; 1: OFF
	2	Input. Scan result D2. 0: ON; 1: OFF
	3	Input. Scan result D3. 0: ON; 1: OFF
	4	Input. Scan result D4. 0: ON; 1: OFF
	5	Input. Scan result D5. 0: ON; 1: OFF
	6	Input. Scan result D6. 0: ON; 1: OFF
	7	Input. Scan result D7. 0: ON; 1: OFF
26	4	Output. Key input interrupt mask 0: Mask 1: Mask open
28	0	Input. Scan result D8
	1	Input. Scan result D9
P15		Input. Key input interrupt flag 0: A keyboard input interrupt has occurred. 1: No keyboard input interrupt has occurred.

Table 2.1: I/O ports related to keyboard input

2.3 Key scan

As shown in Figure 2.1, ten data can be input from each of the eight lines connected to the keyboard. Line L0 inputs data from keys 0, 1, 2, 3, 4, 5, 6, 7, the PF1 key and DIP switch 1. In the same way, data from keys @, A, B, C, D, E, F, G, the PF3 key and DIP switch 3 are input through line L2. This means that to input all of the data from the keyboard, lines L0 to L7 must be selected in turn and the data fetch operation repeated eight times.

In some cases, data may not be input correctly from the keyboard due to this circuit configuration. For example, when keys 1, 8 and 9 are pressed, the circuit recognizes key 0 as having been pressed. There are several such combinations which will cause incorrect data to be received. Key scan is performed by the following procedure:

1. Close key input interrupt mask

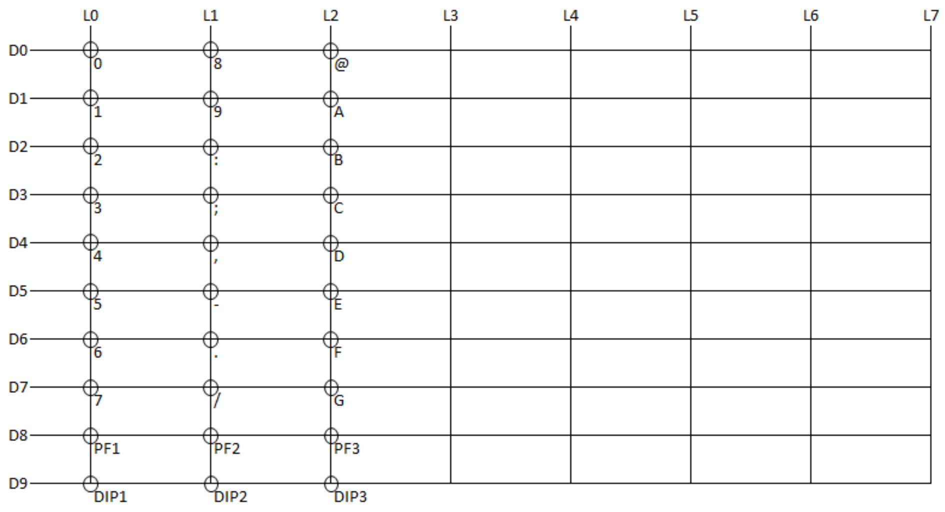


Figure 2.1: Key matrix

P264 (bit 4 at address 26) is an IRQ1 key interrupt mask. As an interrupt occurs if the key is pressed (i.e., the line to scan is specified and the key on the line is pressed) if this mask is open, the interrupt is disabled.

2. Specify line to scan

   There are 8 lines, L0 to L7, and any line can be specified for input. When line L0 is specified, the data on the line L0 can be input. If all lines L0 to L7 are specified, any key can be detected. The bit 0 at address 20 specifies line L0. When the value of this bit is 0, scan is enabled and when 1, scan is not performed. The value FE (line L0 is scanned and the other lines are not scanned) is output first to address 20.

3. Fetch data

   When the contents of the address 22 are input, the data in D0 through D7 can be obtained. When the contents of address 28 are input, the data in D8 and D9 (bit 0, bit 1) can be obtained (there is a wait of several tens of microseconds to obtain correct data after the line is specified). Now, input from keys 0 to 7, PF1 and DIP switch 1 is enabled.

4. Scan lines

   Lines L1 through L7 are sequentially scanned and procedures 2 and 3 above are repeated. In this way all the data from the keyboard as well as the DIP switches values can be input. Table 2.2 shows the arrangement of the key matrix. Figure 2.2 shows the arrangement of the keyboard matrix. The hexadecimal values in Table 2.2 and Figure 2.2 show the correspondence between key layout and positions in the key matrix.

	L0	L1	L2	L3	L4	L5	L6	L7
D0	00$_{0}$	08$_{8}$	10$_{@}$	18$_{\text{H}}$	20$_{\text{P}}$	28$_{\text{X}}$	30$_{\text{RETURN}}$	38$_{\text{CLEAR}}$
D1	01$_{1}$	09$_{9}$	11$_{\text{A}}$	19$_{\text{I}}$	21$_{\text{Q}}$	29$_{\text{Y}}$	31$_{\text{SPACE}}$	39$_{\text{SCRN}}$
D2	02$_{2}$	0A$_{:}$	12$_{\text{B}}$	1A$_{\text{J}}$	22$_{\text{R}}$	2A$_{\text{Z}}$	32$_{\text{TAB}}$	3A$_{\text{BREAK}}$
D3	03$_{3}$	0B$_{;}$	13$_{\text{C}}$	1B$_{\text{K}}$	23$_{\text{S}}$	2B$_{[}$		3B$_{\text{PAUSE}}$
D4	04$_{4}$	0C$_{,}$	14$_{\text{D}}$	1C$_{\text{L}}$	24$_{\text{T}}$	2C$_{]}$		3C$_{\text{DEL}}$
D5	05$_{5}$	0D$_{-}$	15$_{\text{E}}$	1D$_{\text{M}}$	25$_{\text{U}}$	2D$_{\backslash}$	35$_{\text{NUM}}$	3D$_{\text{MENU}}$
D6	06$_{6}$	0E$_{.}$	16$_{\text{F}}$	1E$_{\text{N}}$	26$_{\text{V}}$	2E$_{\rightarrow}$		
D7	07$_{7}$	0F$_{/}$	17$_{\text{G}}$	1F$_{\text{O}}$	27$_{\text{W}}$	2F$_{\leftarrow}$	37$_{\text{CAPS}}$	
D8	40$_{\text{PF1}}$	41$_{\text{PF2}}$	42$_{\text{PF3}}$	43$_{\text{PF4}}$	44$_{\text{PF5}}$	45$_{\text{FEED}}$		
D9	48$_{\text{DIP1}}$	49$_{\text{DIP2}}$	4A$_{\text{DIP3}}$	4B$_{\text{DIP4}}$		4D$_{\text{SHIFT}}$	4E$_{\text{CTRL}}$	4F$_{\text{Printer}}$

Table 2.2: Key matrix

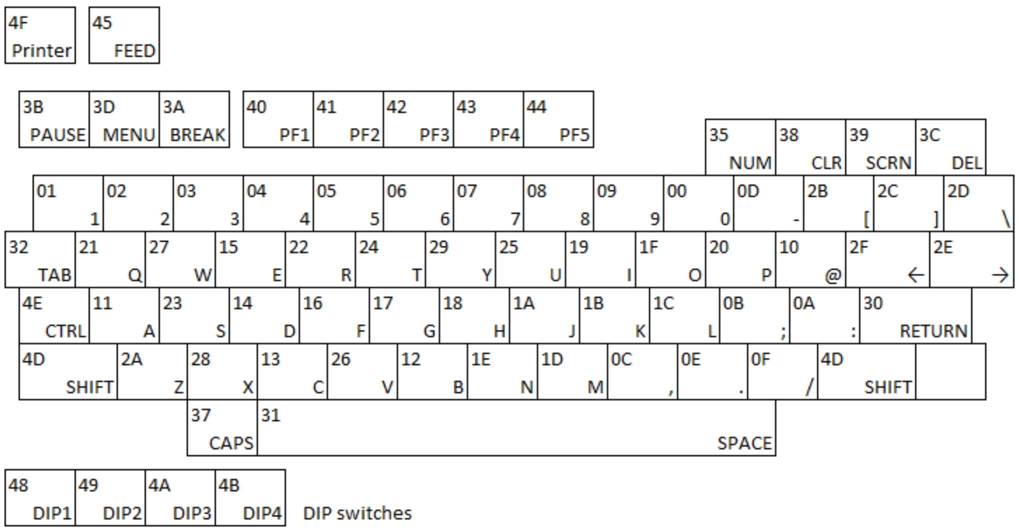


Figure 2.2: Arrangement of the keyboard matrix

2.4 Keyboard input interrupt

An `IRQ1` interrupt is enabled when the keyboard data is input. The following procedure is used to issue an `IRQ1` interrupt.

1. Specify the key line

   The line where an interrupt occurs when a key is pressed is specified. Set '`0`' in address `20` to specify the key scan line. Once '`0`' is set, an interrupt occurs when any key is pressed. Note that the keys and switches on `D9` such as DIP switches 1 to 4, SHIFT keys, CTRL key and Printer ON/OFF switch are excluded from the keyboard input interrupt.

2. Open the keyboard input interrupt mask

   Write '`1`' to bit 5 of address `26` (`P265`) where the keyboard input interrupt mask is performed.

3. Open the CPU interrupt mask

   The CPU interrupt mask is opened by a `CLI` command.

4. Confirm the keyboard interrupt

   If the `P15` is '`0`' when an `IRQ1` interrupt occurs, it indicates the occurrence of the key input interrupt.

2.5 Timing of key input process

An `IRQ1` interrupt occurs when a key is input. Sampling (`OCR` interrupt) is performed using the MCU free running counter.

After a key is pressed as shown in Figure 2.3, the Output Compare Register (`OCR`) issues interrupts at intervals of 20ms (the key interrupt mask is closed) and auto repeat process is performed by key scan. If the same key is pressed continuously for a certain number of key scans after issuance of an `OCF` interrupt, it is assumed that the key has been newly pressed. The received data from the keyboard is stores in the First In, First Out (FIFO) key stack.

2.6 Automatic key input at power ON

The 18-byte variable `KYISTK` contains key input data that can be specified by a monitor `K` command during reset (refer to memory map in Chapter 14).

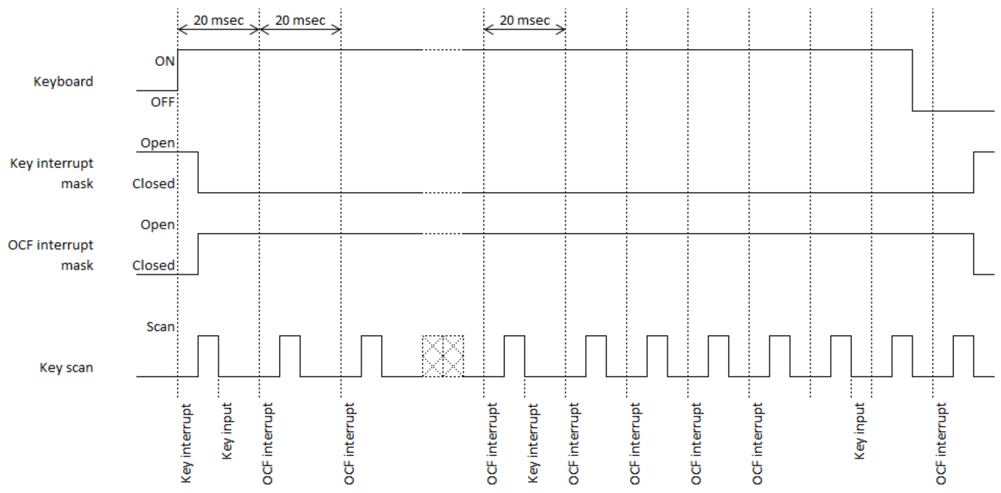


Figure 2.3: Timing of key input

When the value of the variable `KYISFL` is `0A`, the `KYISTK` contains key input data. When the value is `0B`, key input data is currently being fetched from the `KYISTK`. If the value of the `KYISFL` is `0B` when the subroutine `KEYIN` (to fetch the key input data from the key stack) is called, the value obtained from the `KYISTK` is used as the key input data.

2.7 Key input subroutines

The following subroutines for key input are provided.

1. `INITKY`: initializes the keyboard and sets the default value.
2. `KEYSCN`: performs the key scan operation and obtains input data from the pressed key.
3. `KEYIN`: fetches one character from the key input buffer.
4. `KEYSTS`: obtains the number of characters in the key input buffer.
5. `KYSSTK`: specifies the automatic input key data.

2.8 Sleep function

The MCU is provided with a sleep function to reduce power consumption when it is not functioning. The sleep mode is entered during execution of the KEYIN subroutine to wait for key input when the key input buffer is empty.

2.9 Special keys

1. BREAK key

   When the BREAK key is pressed, the data is not taken into the key stack, but the I/O operation is cancelled (subroutine BREKIO is called). Then, the break process is performed to the slave MCU and bit 7 of variables MIOSTS (address 007D) and SIOSTS (address 007C) become ON. The data input from the power ON key stack is cancelled. If bit 7 of the variable RUNMOD (address 007B) is '1', control returns from the key input interrupt. When bit 7 is '0', the subroutine is called starting at the address (0120,0121). The default value of address (0120,0121) is (FF,B2). The subroutine RSTRIO (re-start of I/O operation) is executed at the entry point of the address FFB2 and control jumps to the menu routine. In addition to the BREAK key, the specified subroutines are executed when the MENU, PAUSE, CTRL+PF3, CTRL+PF4 and CTRL+PF5 keys are pressed. Any addresses can be specified.

2. MENU key

   When the MENU key is pressed and bit 7 of the variable RUNMOD is '1', the code FC is input to the key stack and control returns from the interrupt. When bit 7 is '0', control returns from the interrupt after executing the subroutine starting at the address specified by the address (0122,0123).

3. PAUSE key

   When the MENU key is pressed, bit 6 of the variable MIOSTS becomes '1'. Then, control returns from the interrupt if bit 7 od the variable RUNMOD is '1'. If bit 7 is '0', control returns from the interrupt after executing the subroutine starting at the address specified by address (0124,0125).

4. CTRL+PF3, CTRL+PF4 and CTRL+PF5 keys

   When the CTRL+PF3, CTRL+PF4 or CTRL+PF5 keys are pressed, control returns from the interrupt after executing the corresponding subroutine starting at the address specified by the address (0126,0127), (0128,0129) and (012A,012B). If, for example, (FF,10) (the entry point of Monitor) is written to the address (0126,0127), the Monitor will be executed when the CTRL+PF3 keys are pressed.

2.10 Key input modes

The current key input mode (numeric and uppercase, shift, etc.) is indicated by the 1-byte variable KEYMOD. The address of this variable is (FFE4,FFE5). The current data in this address is 0169. Referring to the contents in the address, the current mode (in this case, the keyboard mode) can be recognized. To force-set a certain mode, change the contents of the current address to those of the mode to be set. The following three modes are available.

- Bit 1: numeric mode.
- Bit 2: CAPS mode (lowercase letter mode is assumed when bit 2 is '1').
- Bit 4: graphic mode.

Only one of these bits may be '1' or all of them may be '0'. The current mode is indicated by the bit which is '1'.

2.11 Changing constants

The constants on the RAM are the following.

Key stack size, time interval until the second key input is accepted for auto repeat, time interval until the third or subsequent key input is accepted for auto repeat, sampling interval of key scan and power ON key stack. The default values for these constants are set when the keyboard is initialized. Values set after the default values have been set are used (for details, refer to memory map in Chapter 14).

2.11.1 Key scan

The keyboard value read by key scan is assigned to variable NEWKTB (10 bytes, starting address: (FFD0,FFD1). Table 2.3 shows the format of the keyboard values read by key scan.

In this case, the DIP switches and the Printer ON/OFF switch are set according to the values at address 7F (in other words, software specification takes precedence over the key scan specification).

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
NEWKTB+0	D7(L0)	D6(L0)	D5(L0)	D4(L0)	D3(L0)	D2(L0)	D1(L0)	D0(L0)
NEWKTB+1	D7(L1)	D6(L1)			...			D0(L1)
NEWKTB+2	D7(L2)	D6(L2)			...			D0(L2)
NEWKTB+3	D7(L3)	D6(L3)			...			D0(L3)
NEWKTB+4	D7(L4)	D6(L4)			...			D0(L4)
NEWKTB+5	D7(L5)	D6(L5)			...			D0(L5)
NEWKTB+6	D7(L6)	D6(L6)			...			D0(L6)
NEWKTB+7	D7(L7)	D6(L7)			...			D0(L7)
NEWKTB+8	D8(L7)	D8(L6)	D8(L5)	D8(L4)	D8(L3)	D8(L2)	D8(L1)	D8(L0)
NEWKTB+9	D9(L7)	D9(L6)	D9(L5)	D9(L4)	D9(L3)	D9(L2)	D9(L1)	D9(L0)

Table 2.3: Key scan values

2.12 Keyboard input subroutines

Name of subroutine	Entry point	Description
INITKY	FFA0	Initializes key input. Sets the initial values (including default values). Specifies the vectors jumped to when the BREAK, MENU, PAUSE, CTRL+PF3, CTRL+PF4 and CTRL+PF5 keys are pressed. Specifies default values for timing such as sampling time, etc. Specifies the number of key stack data, and the key stack used when the power is turned ON
		• Parameters: – At entry: none – At return: none • Registers retained: none • Subroutines referenced: none • Variables used: none
Continues in next page...		

...continued from previous page.

Name of subroutine	Entry point	Description
`KEYSTS`	`FF9D`	Inputs the number of key stack data (excluding the key stack used when the power is turned ON).
		• Parameters: – At entry: none – At return: * `(C)`: abnormal I/O flag. * `(A)`: number of key stack data (in bytes). * `(Z)`: according to the value of `(A)`. • Registers retained: `(B)` and `(X)` • Subroutines referenced: none • Variables used: none **Note:** when a `PF` key is pressed, the number of stack data increases by 2.
`KEYIN`	`FF9A`	Inputs one character from the key stack. If no data exists in the key stack, this subroutine lets the MCU sleep and waits until data is input from the keyboard. If data exists in the key stack when the power is turned ON, this data is recognized as input data. If both key stack data and keyboard data are available, the key stack data take precedence over keyboard data.

Continues in next page...

...continued from previous page.

Name of subroutine	Entry point	Description
		• Parameters: – At entry: none – At return: ∗ `(C)`: abnormal I/O flag. ∗ `(A,B)`: character code. 1-byte codes are stored in `(A)` and 2-byte codes (`(A)`=`FE`) are stored in `(A,B)`. • Registers retained: `(X)` • Subroutines referenced: none • Variables used: none

`KEYSCN`	`FF6A`	Scans the key matrix. The result of the key scan is stored in `NEWKTB` (10 bytes). Note that the DIP switches and Printer ON/OFF switch are set according to the value of the variable `SDIPS2`. The contents of `NEWKTB` are:
		
		`          Bit7 Bit6 ··· Bit0`
		`NEWKTB+0  D7   D6   ··· D0   ··· L0`
		`NEWKTB+1  D7   D6   ··· D0   ··· L1`
		`  ⋮       ⋮    ⋮    ⋱   ⋮        ⋮`
		`NEWKTB+8  L7   L6   ··· L0   ··· D8`
		`NEWKTB+9  L7   L6   ··· L0   ··· D9`

Continues in next page...

...continued from previous page.		
Name of subroutine	Entry point	Description
		• Parameters: – At entry: none – At return: none • Registers retained: none • Subroutines referenced: none • Variables used: `K0` and `K1` (the values of these variables are retained).
`KYSSTK`	`FF22`	Inputs data to the key stack when the power is turned ON. The size of the key stack is 18 bytes. If more than 18 bytes of data are input, the excess data is ignored.
		• Parameters: – At entry: ∗ `(X)`: starting address of character strings. ∗ `(B)`: number of characters (0 to 18: the key stack is cleared when the number is 0). – At return: none • Registers retained: none • Subroutines referenced: none • Variables used: none

Table 2.4: Keyboard input subroutines

2.13 Keyboard work area

Address (from)	(to)	Variable name	Bytes	Description
0140	0140	KSTKSZ	1	The size of the key stack. The default value is 8. May be specified in the range 1 to 15. If the value is '1', input of PF keys is not accepted.
0141	0141	KICNT1	1	Time until the first key input is accepted for auto repeat. The unit depends on sampling time. The default value for sampling time is 20ms. The default value is 40_{10} (800ms).
0142	0142	KICNT2	1	Time until the second or subsequent key input is accepted for auto repeat. The units are the same as those of KICNT1. The default value is 6 (120ms).
0143	0144	KICNTM	2	Sampling time. The unit 1 equals approximately 1.6 µs. The default value is 12288_{10} (20ms).
0145	014E	NEWKTB	10	Value of the key scan matrix. The status after the key scan is stored in this area. '1' denotes the ON condition. Bit 0 at the first address of the work area corresponds to 00 of the key matrix table and bit 7 corresponds to 07. In this manner, bit 7 of the last address corresponds to 4F.
014F	0158	OLDKTB	10	Value of the previous key scan. The previous value of NEWKTB is stored in this variable.
0159	0162	CHKKTB	10	Stores the data for the position of the newly pressed key after key scan.
0163	0164	KYISAD	2	Address of automatic key input when power is turned ON. Set to 016F at reset.

Continues in next page...

...continued from previous page.

Address (from)	(to)	Variable name	Bytes	Description
0165	0165	KYISFL	1	Flag indicating whether or not data exists in the key stack when power is turned ON. When this flag is `0A`, data exists in the key stack but the fetch operation ends. When the flag is `0B`, data is currently being fetched from the stack. If the flag is other than `0A` and `0B`, no data exists in the key stack.
0166	0166	KYISCN	1	The number of data in the automatic input key stack. Value is in the range 0 to 255_{10}.
0167	0167	KYISPN	1	The number of data input from the automatic key input. The number is in the range 0 to the value specified by `KYISCN`.
0168	0168	KYISPN	1	The number of data in the input key stack. The number is in the range 0 to the value specified in `KSTKSZ`.

Continues in next page...

...continued from previous page.

Address (from)	(to)	Variable name	Bytes	Description
0169	0169	KEYMOD	1	Input key modes. This address indicates the uppercase, numeric modes, etc. • Bit 1: numeric mode when this bit is '1'. • Bit 2: lowercase mode when this bit is '1'. • Bit 3: unused. • Bit 4: graphic mode when this bit is '1'. • Bit 5: SHIFT mode when this bit is '1'. • Bit 6: the CTRL key has been pressed when this bit is '1'. • Bit 7: indicates that a special key such as the BREAK, PAUSE, MENU or one of the PF keys has been pressed when this bit is '1'. One of bits 0 through 4 must be ON or all bits must be OFF.
016A	016A	ONKFLG	1	Indicates the key input status: • 00: inhibits key reception. Waits until the pressed key is released. • FF: key input enabled. • 01: auto repeat function.

Continues in next page...

...continued from previous page.				
Address (from)	(to)	Variable name	Bytes	Description
016B	016B	KPRFLG	1	For auto repeat, this variable indicates the number of times the same key input has been received. When this value equals that of KICNT1 or KICNT2, the pressed character is taken to be input once.
016C	016C	KEYRPT	1	Indicates the auto repeat key position on the matrix. Refer to Table 2.2.
016D	016E	CKEYRD	2	Input key code. A PF key is 2 bytes.
016F	0180	KYISTK	18	Work area for the power ON key stack.
0181	0188	CHRSTK	8	Work area for the key stack.
0189	018F		7	This area is secured for the expansion of the key stack.

Table 2.5: Keyboard work area

Chapter 3

Liquid crystal display (LCD)

3.1 General

The liquid crystal display (LCD) has a resolution of 120 horizontal dots and 32 vertical dots and LCD controllers which enable the specification of data for each dot.

6 LCD controllers together control the LCD, each of which controls an area of 40 horizontal by 16 vertical dots.

Normally, a single character is displayed in a matrix of 6 horizontal by 8 vertical dots. Alphanumeric characters, however, are actually formed in a matrix 5 by 7 dots as spaces are left between characters on the screen.

3.2 Functions of LCD controllers

As above mentioned, the 6 controllers together control the LCD. The dot areas controlled by each controller are shown in Table 3.1.

As shown in Table 3.1, each controller is responsible for an area of 40 by 16 dots.

Each controller has data addresses `0` to 27_{16} in the row direction. Data consists of 8 bits (Figure 3.1).

3.3 I/O ports for display and input

See Table 3.2.

	0 39	40 79	80 119
0 7	Controller 1 (bank 0)	Controller 2 (bank 0)	Controller 3 (bank 0)
8 15	Controller 1 (bank 1)	Controller 2 (bank 1)	Controller 3 (bank 1)
16 23	Controller 4 (bank 0)	Controller 5 (bank 0)	Controller 6 (bank 0)
24 31	Controller 4 (bank 1)	Controller 5 (bank 1)	Controller 6 (bank 1)

Table 3.1: Dot area controlled by each LCD controller

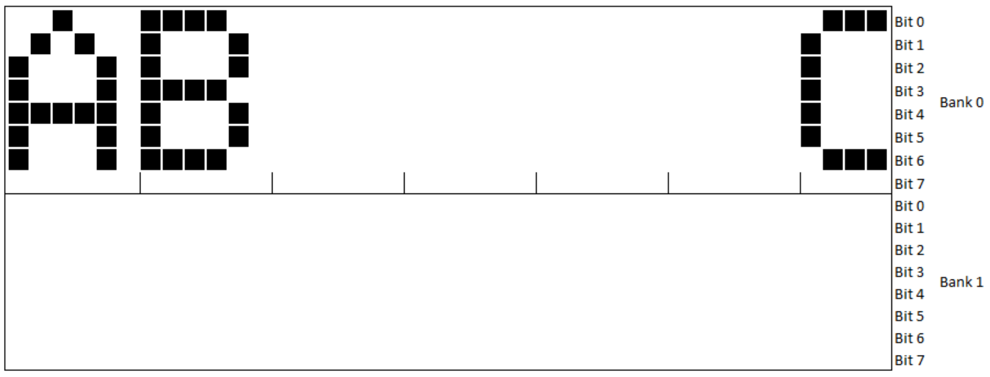


Figure 3.1: Displayed contents of each LCD controller

MCU	Address	Bit position	Description
Master	26	0-2	Output: selection of LCD driver • 1-6: controllers 1 to 6 selected • 0: no controller is selected
		3	Output: selection of data or command for LCD driver • 0: data • 1: command

Continues in next page...

...continued from previous page

MCU	Address	Bit position	Description
	28	7	Input: BUSY signal of LCD controller • 0: busy
	2A		Output: serial clock to LCD controller
	2B		Output: serial clock to LCD controller
	2A	0	Output: output data to LCD controller
		1	Output: output data to LCD controller
		2	Output: output data to LCD controller
		3	Output: output data to LCD controller
		4	Output: output data to LCD controller
		5	Output: output data to LCD controller
		6	Output: output data to LCD controller
		7	Output: output data to LCD controller

Table 3.2: I/O ports related to LCD controllers controller

3.4 Data display procedure

Data is displayed on the LCD by the following procedure.

1. Selection of controller

   One of the 6 controllers is selected by specifying an appropiate value in the bits 0 to 2 of address `26` using subroutine `WRTP26`. If `0` is specified, no controller is selected. System default `0` is for power conservation.

2. Command setting

   The bit 3 of address `26` is a bit used to select either data or command for the selected controller. When this bit is set to '`1`', a command is selected. This data/command selection may be performed simultaneously with the controller selection described in 1 above. Set a command in address `2A` and confirm that the LCD controller is ready (when the bit 7 of address `28` is '`1`'). Then apply 8 serial clock pulses to the controller. Address `2A` is read 8 times. Since address `2B` is also for serial clock input, 8 serial clock pulses are given to the controller upon execution of "`LDD $2A`" 4 times.

3. Data

   When the bit 3 of address `26` is set to '`0`', data is selected. The data setting procedure is the same as the command setting described above. Depending on the type of command, data must be continuously output for display.

3.5 Input of display data

The bit indicating that the controller is busy (i.e., bit `7` of address `28`) becomes the input data for display.

3.6 Display subroutines

The HX-20 has the following three subroutines for character display:

1. `DSPLCN`: displays n characters of data (ASCII code) on the physical screen.

2. `DSPLCH`: displays one character of data (ASCII code) on the physical screen.

3. `DISPIT`: displays one character of data (ASCII code) on the physical screen.

3.7 Coordinates on the LCD

(x, y) indicates the coordinates on the LCD. x is the coordinate in the horizontal direction (columns) and y is the coordinate in the vertical direction (rows). $(0, 0)$ indicates that the positions of both the vertical and the horizontal coordinates are the upper left edge of the LCD. The values of x, y coordinates on the text screen must be in the range shown below.

$$0 \leq x \leq 19 \text{ and } 0 \leq y \leq 3$$

The values of x, y coordinates on the graphic screen must be in the range shown below.

$$0 \leq x \leq 119 \text{ and } 0 \leq y \leq 31$$

3.8 LCD subroutines

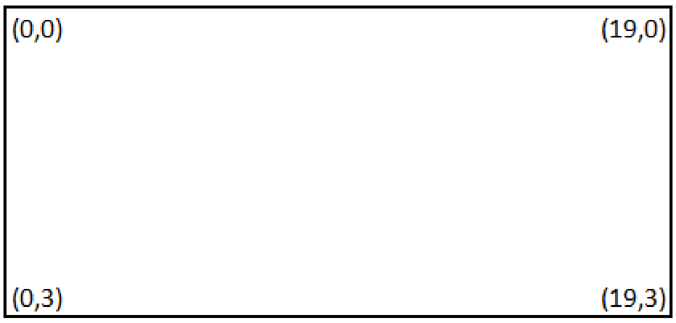


Figure 3.2: Coordinates on the LCD (text screen)

Name of subroutine	Entry point	Description
`DSPLCN`	`FF49`	Displays or clears n characters on the physical screen.

Continues in next page...

...continued from previous page.

Name of subroutine	Entry point	Description
		• Parameters: – At entry: ∗ `(B)`: number of characters displayed. The screen is cleared when `(B)` is 0. ∗ `(X)`: starting address of data packet. This parameter need not be specified when `(B)` is 0. ∗ Data packet · Byte 0: x coordinate (0 to 19_{10}) of the display position of the first character · Byte 1: y coordinate (1 to 3_{10}) of the display position of the first character · Byte 2: character code (ASCII) · Byte 3: character code (ASCII) · Byte $n + 1$: character code (ASCII) – At return: none • Registers retained: none • Subroutines referenced: `DSPLCH` • Variables used: none
`DSPLCH`	`FF4C`	Displays one character on the physical screen. The display data is first written into the screen `PSBUF`.

Continues in next page...

...continued from previous page.

Name of subroutine	Entry point	Description
		• Parameters: – At entry: * `(A)`: ASCII character code * `(X)`: display position on LCD. (*high*, *low*) = x coordinate (0 to 19_{10}), y coordinate (0 to 3_{10}). – At return: none • Registers retained: `(A)`, `(B)` and `(X)`. • Subroutines referenced: `CHRGEN`, `LCDMOD` and `DATMOD`. • Variables used: none
`DISPIT`	`FF5B`	Displays one character on the physical screen. The display data is not written into the screen `PSBUF`.
		Parameters at entry and return, registers retained, subroutines referenced and variables used are the same as those for subroutine `DSPLCH`
`CHRGEN`	`FF67`	Generates the character pattern. A character pattern (6×8 dots) is provided for display of the character specified by the ASCII code on the LCD. Certain character patterns may change according to the value set by the DIP switch for different countries.

Continues in next page...

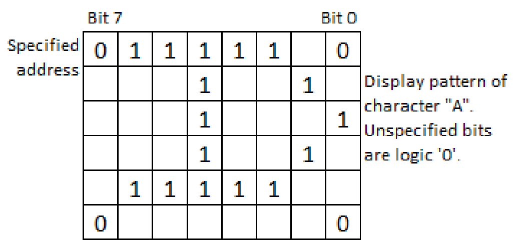
...continued from previous page.		
Name of subroutine	Entry point	Description
		• Parameters: – At entry: ∗ `(A)`: character code ∗ `(X)`: starting address where 6-byte character display pattern is stored. – At return: character display pattern (specified address)  Bit 7 Bit 0 Specified address 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 Display pattern of character "A". Unspecified bits are logic '0'. • Registers retained: none • Subroutines referenced: none • Variables used: none • Others: re-entrant

Table 3.3: LCD subroutines

3.9 Screen routines work area

Address (from)	Address (to)	Variable name	Bytes	Description
0220	026F	PSBUF	80	Positions of data (ASCII codes) displayed on the physical screen represented in $(column, row)$ format as follows: $$\begin{array}{cccc} (0,0) & (1,0) & \cdots & (19_{10},0) \\ \vdots & \vdots & \ddots & \vdots \\ (0,3_{10}) & (1,3_{10}) & \cdots & (19_{10},3_{10}) \end{array}$$
0270	0271	SCRTOP	2	Starting address of the virtual screen buffer
0272	0273	SCRBOT	2	Ending address of the virtual screen buffer
0274	0275	DISTOP	2	Starting position of the physical screen on the virtual screen. The address of position $(0,0)$ of the physical screen in the physical screen buffer.
0276	0276	VSCRX	1	Virtual screen width indicated as the maximum values of x coordinate
0277	0277	VSCRY	1	Virtual screen length indicated as the maximum values of y coordinate
0278	0278	CURX	1	x coordinate of the cursor position (on the physical screen)
0279	0279	CURY	1	y coordinate of the cursor position (on the physical screen)
027A	027A	LRMODE	1	Scroll step x (left and right)
027B	027B	UDMOD	1	Scroll step y (up and down)
027C	027C	CURMRG	1	Scroll margin (1 through 10_{10})
027D	027D	SSPEED	1	Vertical scrolling speed (0 to 9). When the scrolling speed is set at 8, there is a 130ms wait between each scroll. This wait is increased in 130ms increments for each setting: 7, 6, 5,... 0. When a scrolling speed of 9 is specified, there is no wait between vertical scrolls.

Continues in next page...

...continued from previous page.

Address (from)	(to)	Variable name	Bytes	Description
`027E`	`027E`	`DISPX`	1	x coordinate (0 to 19_{10}) of the character to be displayed on the physical screen by a virtual screen routine
`027F`	`027F`	`DISPY`	1	y coordinate (0 to 3) of the character to be displayed on the physical screen by a virtual screen routine

Continues in next page...

...continued from previous page.

Address (from)	(to)	Variable name	Bytes	Description
`0280`	`0280`	`DISSTS`	1	Indicates the display status. • Bit 0: indicates whether or not left and right scrolling is permitted. – `1`: scrolling disabled. – `0`: scrolling enabled. • Bits 1-3: not used. • Bit 4: indicates whether or not there is a wait in vertical scrolling. – `1`: wait executed. – `0`: no wait executed. • Bit 5: Cursor ON/OFF switch. Determines whether the cursor ('_' below the character) will be displayed on the physical screen. – `1`: cursor ON. – `0`: cursor OFF. • Bit 6: indicates cursor ON/OFF status. – `1`: cursor ON. – `0`: cursor OFF.

Continues in next page...

...continued from previous page.				
Address (from)	(to)	Variable name	Bytes	Description
				• Bit 7: flag to indicate whether or not the entire screen is to be rewritten. – `0`: display for only one character. – `1`: rewrites the entire screen. **Note:** all screen data must be checked and rewritten even if only one character is to be displayed. However, since doing this adversely affects operating speed, this switch is used to reduce the amount of screen data checked and rewritten. Bits 5 and 6 The following two operations are required to display a character at the cursor position and then move the cursor to the next position: 1. Display the character at the cursor position The cursor is turned OFF and the character is displayed at the cursor position. 2. Cursor movement A space character is displayed where the cursor is ON. Bit 5 is used to control cursor ON/OFF condition and bit 6 determines whether the cursor will be displayed on the screen.
`0281`	`0285`		5	Used as temporary work area
`0286`	`028B`	`CHRPTN`	6	Contains the character font (result of subroutine `CHRGEN`)

Table 3.4: Screen work area

3.10 Sample listings: LCD driver routine

```
   0 :                    ; LCD driver routine
   0 :                    ;
   0 :                    ; I/O port
   0 :                    ;   $28
   0 :                    ;     7:R LCD driver busy
   0 :                    ;       0: busy; 1: ready
   0 :                    ;   $26
   0 :                    ;     0:W LCD command/data 1
   0 :                    ;     1:W LCD command/data 2
   0 :                    ;     2:W LCD command/data 3
   0 :                    ;     3:W LCD command/data selection
   0 :                    ;       0: data; 1: command
   0 :                    ;     4:W keyboard interrupt mask
   0 :                    ;       0: close; 1: open
   0 :                    ;     5:W peripheral control (to serial)
   0 :                    ;     6:W to plug in 1
   0 :                    ;     7:W to plug in 2 and slave P40
   0 :                            PAGE   0
   0 :                            CPU    6301
   0 :                    ;
   0 :                    ; Subroutine entry point
   0 : =$FED4             WRTP26 EQU    $FED4
   0 : =$FF67             CHRGEN EQU    $FF67
   0 :                    ; Work area
   0 : =$7D               MIOSTS EQU    $7D  ; Main I/O status
   0 :                                       ; Bit 0: on read/write to LCD 1: reading/writing...
   0 : =$286              CHRPTL EQU    $286 ; Work area to store character pattern
   0 : =$220              PSBUF  EQU    $220 ; Character codes on physical screen
   0 : =$280              DISSTS EQU    $280 ; Display status
   0 :                                       ; Bit 5: cursor on with character pattern fl...
   0 :                                       ;         (1:on)
   0 :                    ;
   0 :                      ;
1000 :                            ORG    $1000
1000 :                      ; Display one character to real LCD screen
1000 :                      ; routine 'DISPIT': display 1 character to LCD without buffer
1000 :                      ;         'DISPCH': display one character to LCD and write to buffer
1000 :                      ; On entry
1000 :                      ;   (A): ASCII character code
1000 :                      ;   (X): LCD position (high:column, low:line)
1000 :                      ; On exit
1000 :                      ;   (X): next data position (if illegal addressing, change to legal)
1000 :                      ;
1000 :                      ;
1000 :                      ; Check display position and generate character font
1000 :                      ; On entry
1000 :                      ;   (X): display address (high byte:column, low byte:line)
1000 :                      ;   (A): displayed character
1000 :                      ; On exit
1000 :                      ;   (A,B): modified position
1000 :                      ; Register preseve (X) <- (A,B)
1000 : 72 01 7D             DPCHEK OIM    #$1,MIOSTS    ; Set flag LCD is on writing
1003 : 37                          PSHB
1004 : 36                          PSHA
1005 : 3C                          PSHX
1006 : CE 02 86                    LDX    #CHRPTL       ; Set character pattern to 'CHRPTL'
1009 : BD FF 67                    JSR    CHRGEN
100C : 32                          PULA
100D : 33                          PULB
100E : 38                          PULX                 ; Note. (X) <-> (A,B)
100F : 81 13                       CMPA   #19           ; Is column limit out of range?
1011 : 25 02                       BCS    NONOVX        ; No.
1013 : 86 13                       LDAA   #19           ; Yes. Limit=19
1015 : C4 03                NONOVX ANDB   #3
```

```
1017 :                ;
1017 : 39                    RTS
1018 :                ;
1018 :                ; Display one character to real screen without writing to screen buffer
1018 :                ; On entry
1018 :                ;   (A): character (ASCII code)
1018 :                ;   (X): display address (high:column, low:line)
1018 :                ;
1018 :                ; Entry point
1018 : 8D E6          DISPIT BSR    DPCHEK
101A : 3C                    PSHX                 ; Save value of (A,B)
101B : 20 15                 BRA    NONSET
101D :                ; Entry point
101D : 8D E1          DSPLCH BSR    DPCHEK
101F :                ;
101F : 3C                    PSHX                 ; Save (A,B)
1020 : 37                    PSHB
1021 : 36                    PSHA
1022 : 86 14                 LDAA   #20
1024 : 3D                    MUL
1025 : 30                    TSX
1026 : EB 00                 ADDB   0,X           ; Address offset <- (B)*width + (A)
1028 : A6 02                 LDAA   2,X
102A : CE 02 20              LDX    #PSBUF        ; (X) <- physical screen buffer address
102D : 3A                    ABX
102E : A7 00                 STAA   0,X
1030 : 32                    PULA
1031 : 33                    PULB
1032 :                ; Calculate address in LCD driver
1032 : 37             NONSET PSHB                 ; Save location pointer (X,Y)
1033 : 36                    PSHA
1034 :                ;                           ; Already saved four bytes
```

```
1034 :                ;                          ; Stack+0: column    Stack+1: line
1034 :                ;                          ; Stack+2: (A)       Stack+3: (B)
1034 : 37                    PSHB
1035 : 48                    ASLA                ; (A) <- (A) * 6 (dot column)
1036 : 16                    TAB
1037 : 48                    ASLA
1038 : 1B                    ABA
1039 : 33                    PULB
103A :                ;
103A :                ; Work use stack
103A :                ;   Stack     00: dot counter 1 (1 character = 6 dot lines)
103A :                ;             01,02: character font address
103A :                ;             03,04: dot column, line
103A :                ;
103A : 37                    PSHB
103B : 36                    PSHA
103C :                ;
103C :                ; ***** LCD drive routine ***** (X):character pattern top address
103C :                ; Set character to driver
103C : CE 02 86              LDX    #CHRPTL
103F : 3C                    PSHX
1040 : 5F                    CLRB
1041 : 37                    PSHB
1042 : 30                    TSX
1043 : EC 03          DISCHL LDD    3,X
1045 : 8D 56                 BSR    LCADDR       ; Get chip no. & data address.(DATADD,CHIPNO)
1047 : 16                    TAB                 ; Save LCD address to (B)
1048 : 86 64                 LDAA   #$64         ; Select write mode
104A : 8D 76                 BSR    LCDMOD       ; Set command
104C : 17                    TBA
104D : 8A 80                 ORAA   #$80         ; Set data addr to LCD driver.(auto increment)
104F : 8D 71                 BSR    LCDMOD
```

```
1051 : CC 08 00              LDD    #$0800        ; Set data mode code for 'WRTP26' routine
1054 : BD 10 DC              JSR    DATMOD        ; LCD driver: enter data mode (not command)
1057 : E6 00          WRTLOP LDAB   0,X           ; Get 8 bits pattern
1059 : EE 01                 LDX    1,X           ; (B): dot position (0 - 5)
105B : 3A                    ABX
105C : A6 00                 LDAA   0,X
105E : C1 05                 CMPB   #5            ; Last dot (6th): without cursor
1060 : 27 09                 BEQ    DISC20
1062 : F6 02 80              LDAB   DISSTS        ; Cursor on?
1065 : C5 20                 BITB   #$20
1067 : 27 02                 BEQ    DISC20
1069 : 8A 80                 ORAA   #$80
106B : 8D 55          DISC20 BSR    LCDMOD        ; Write one byte bit pattern
106D : 30                    TSX
106E : 6C 00                 INC    0,X           ; Complete to write 6 bytes?
1070 : A6 00                 LDAA   0,X
1072 : 81 06                 CMPA   #6
1074 : 27 0E                 BEQ    ENDDIC        ; Yes. End
1076 : 6C 03                 INC    3,X
1078 : A6 03                 LDAA   3,X           ; Increment data address
107A : 81 28                 CMPA   #40           ; Boundary of driver = 40
107C : 27 C5                 BEQ    DISCHL
107E : 81 50                 CMPA   #80
1080 : 27 C1                 BEQ    DISCHL        ; Chip last add?
1082 : 20 D3                 BRA    WRTLOP        ;
1084 :                ;
1084 :                ;
1084 : CC 0F 08       ENDDIC LDD    #$0F08        ; Chip select off. Command mode
1087 : 8D 53                 BSR    DATMOD        ;
1089 : 31                    INS                  ; Recover stack pointer (+5)
108A : 38                    PULX
108B : 38                    PULX
```

```
108C : 32                  PULA                   ; Restore position on LCD
108D : 33                  PULB
108E : 4C                  INCA
108F : 81 14               CMPA   #20             ; Next pointer
1091 : 26 04               BNE    DIC100
1093 : 4F                  CLRA
1094 : 5C                  INCB
1095 : C4 03               ANDB   #$03
1097 : 71 FE 7D     DIC100 AIM    #$FF-$1,MIOSTS; LCD flag LCD not busy
109A : 38                  PULX
109B : 18                  XGDX                   ; Return next display point
109C : 39                  RTS
109D :              ;
109D :              ;
109D :              ; Select LCD driver and calculate bank and address pointer
109D :              ; Note: set to $26 driver address, but not set to LCD driver, only return
109D :              ;       LCD address to (A).
109D :              ; On entry
109D :              ;   (A): dot line column position (00 - 119)
109D :              ;   (B): line (0 - 3)
109D :              ; On exit
109D :              ;   (A): dot pointer in the LCD driver
109D :              ;   (B): chip no. (bit3=1)
109D :              ; Register preserve (X)
109D :              ;
109D : 3C           LCADDR PSHX                   ; Save (X)
109E : 37                  PSHB                   ; Stack value of (B)
109F : 30                  TSX
10A0 : 5F                  CLRB
10A1 : 80 28        LCAD10 SUBA   #40             ; (A) <- address and bank
10A3 : 5C                  INCB                   ; (B) <- chip no.
10A4 : 24 FB               BCC    LCAD10
```

```
10A6 : 8B 28                   ADDA   #40           ; Get start address. (B): 1-3
10A8 : 6B 01 00                TIM    #$1,0,X       ; Check bank (odd line no. = bank 1)
10AB : 27 02                   BEQ    LCAD20
10AD : 8A 40                   ORAA   #$40
10AF : 6B 02 00         LCAD20 TIM    #$2,0,X       ; Check driver chip (line >=2, 4-6)
10B2 : 27 02                   BEQ    LCAD30
10B4 : CB 03                   ADDB   #3            ; Chip is 4, 5 or 6
10B6 : 31               LCAD30 INS
10B7 : 38                      PULX
10B8 : 36                      PSHA
10B9 : CA 08                   ORAB   #$08          ; Bit3= data mode bit (1:command)
10BB : 86 0F                   LDAA   #$0F          ; Set chip no.
10BD : BD FE D4                JSR    WRTP26        ; Selected driver chip, and enter command mode
10C0 :                  ;
10C0 : 32                      PULA                 ; Set data address to driver
10C1 : 39                      RTS
10C2 :                  ;
10C2 :                  ;
10C2 : 37               LCDMOD PSHB
10C3 : 16                      TAB
10C4 : 36                      PSHA
10C5 : 07                      TPA
10C6 : 0F                      SEI
10C7 : 3C                      PSHX
10C8 : D7 2A                   STAB   $2A           ; Set add or mode
10CA : 7D 00 28         LCDM10 TST    $28           ; 7 bit is LCD busy flag
10CD : 2A FB                   BPL    LCDM10        ; Wait
10CF : DE 2A                   LDX    $2A           ; LDD send 2 pulses, so 2 bit shift
10D1 : DE 2A                   LDX    $2A
10D3 : DE 2A                   LDX    $2A
10D5 : DE 2A                   LDX    $2A
10D7 : 38                      PULX
```

```
10D8 : 06                     TAP
10D9 : 32                     PULA
10DA : 33                     PULB
10DB : 39                     RTS
10DC :                 ; After check LCD busy, call 'WRTP26'
10DC :                 ; On entry, (same as 'WRTP26')
10DC :                 ;   (A): target bit position
10DC :                 ;   (B): data
10DC :                 ;
10DC : 7D 00 28        DATMOD TST    $28           ; 7 bit is LCD busy flag
10DF : 2A FB                  BPL    DATMOD        ; Wait
10E1 : 7E FE D4        DTMD10 JMP    WRTP26        ; Set interrupt mask
10E4 :                 ;
10E4 :                 ;
10E4 :                 ; Initialize LCD routine
10E4 :                 ;   Driver initialize and clear cursor on flag
10E4 :                 ; On entry parameter none
10E4 :                 ;
10E4 : 86 10           INITLC LDAA   #$10          ; SFF (Set Frame Frequency) command
10E6 : 8D 0F                  BSR    STRALL        ; Set it for each driver
10E8 : 86 1E                  LDAA   #$1E          ; ACCA : SMM (Set Multiplexing Mode) command
10EA : 8D 0B                  BSR    STRALL        ; Note: 1st driver: SMM value=$1E
10EC :                 ;                           ;       2nd - 6th driver: SMM=$1C
10EC : 86 08                  LDAA   #$08          ; ACCA: disp OFF command. ACCB: chip no.
10EE : 8D 07                  BSR    STRALL
10F0 :                 ;
10F0 : 7F 02 80               CLR    DISSTS        ; Clear display status for non cursos clear
10F3 :                 ;
10F3 : 8D 18                  BSR    LCDCLR        ; Clear screen
10F5 :                 ;
10F5 : 86 09                  LDAA   #$09          ; Display ON command
10F7 :                 ;
```

```
10F7 :                ;
10F7 :                ;****** Set command all drivers
10F7 :                ; Set command to LCD driver
10F7 :                ; On entry
10F7 :                ;   (A): command for LCD driver
10F7 :                ; On exit
10F7 :                ; Register preserve X
10F7 :                ; Note: this routine must be call only 'INITLC'
10F7 :                ;       because command will be changed
10F7 :                ;
10F7 : 5F             STRALL CLRB                ; (B): driver number
10F8 : 5C             STRA10 INCB
10F9 : 37                    PSHB
10FA : CA 08                 ORAB   #$08
10FC : 36                    PSHA
10FD : 86 0F                 LDAA   #$0F         ; Select driver and command mode
10FF : BD FE D4              JSR    WRTP26
1102 : 32                    PULA
1103 : 8D BD                 BSR    LCDMOD       ; Set command to driver
1105 : 84 FD                 ANDA   #$FF-$2      ; To change '1E' (SMM command) to '1C'
1107 : 33                    PULB                ; Other commands are $10, $08, $09 (no effect)
1108 : C1 06                 CMPB   #6
110A : 26 EC                 BNE    STRA10
110C : 39                    RTS
110D :                ;
110D :                ;
110D :                ; Clear LCD screen
110D :                ; On entry
110D :                ;   Parameter none
110D :                ; On exit
110D :                ;   (X): 0
110D :                ; Register preserve none
```

```
110D :                 ;
110D : CE 00 00        LCDCLR LDX    #0            ; Pointer set
1110 : 86 20           LCDC10 LDAA   #$20          ; Set space code
1112 : BD 10 1D               JSR    DSPLCH
1115 : 08                     INX                  ; IX has display pointer
1116 : 09                     DEX
1117 : 26 F7                  BNE    LCDC10        ; Not end
1119 : 39                     RTS
111A :                 ;
111A :                 ;
111A :                 ; Display character string to LCD
111A :                 ; On entry
111A :                 ;   (B): number of character string (0 - 80)
111A :                 ;   (X): address of data packet
111A :                 ;        Data packet: (address X), (address Y), Data1,...
111A :                 ; On exit
111A :                 ;   Parameter none
111A :                 ; Register preserve none
111A :                 ; Enable reentrant
111A :                 ; Work use: Stack: Stack + 0,1: Location of character in LCD
111A :                 ;                          2,3: Address of stored character
111A :                 ;                          4  : displayed character number
111A : 5D              DSPLCN TSTB                  ; If (B)=0, clear screen
111B : 27 F0                  BEQ    LCDCLR
111D : 37                     PSHB
111E : 3C                     PSHX
111F : EE 00                  LDX    0,X            ; Get location of display
1121 : 3C                     PSHX                  ;
1122 : 5F                     CLRB                  ; Counter of displayed character
1123 : 30                     TSX
1124 : EE 02           DSPL10 LDX    2,X
1126 : 3A                     ABX
```

```
1127 : A6 02                  LDAA   2,X         ; (A) <- displayed character
1129 : 38                     PULX               ; (X): location on LCD
112A : BD 10 1D               JSR    DSPLCH      ; Display one character to screen
112D : 3C                     PSHX
112E : 5C                     INCB
112F : 30                     TSX
1130 : E1 04                  CMPB   4,X         ; Finished?
1132 : 26 F0                  BNE    DSPL10
1134 : 38                     PULX
1135 : 38                     PULX
1136 : 33                     PULB
1137 : 39                     RTS
1138 :                 ;
1138 :                        END
```

Chapter 4

Serial communication

4.1 General

Serial communication is performed by the start-stop synchronous transmission system. In start-stop transmission, the signal is logic `1` while no data is being sent and becomes `0` to show the start of data (see Figure 4.1).

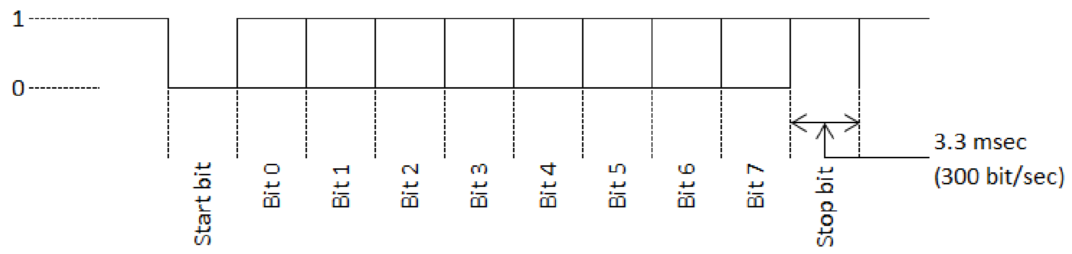


Figure 4.1: Start-stop data transmission format

Figure 4.2 shows an example of signal status when data `3A` (00111010_2) is transmitted in a start-stop format with a single stop bit.

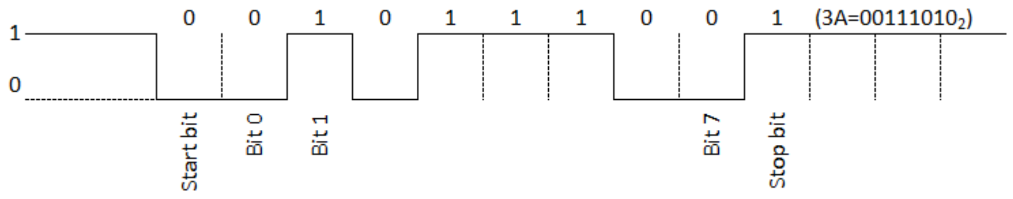


Figure 4.2: Start-stop data `3A`

Data `1` is represented by a low signal (−3 to −8V) and data `0` by a high signal (+3 to +8V) as shown in Figure 4.3.

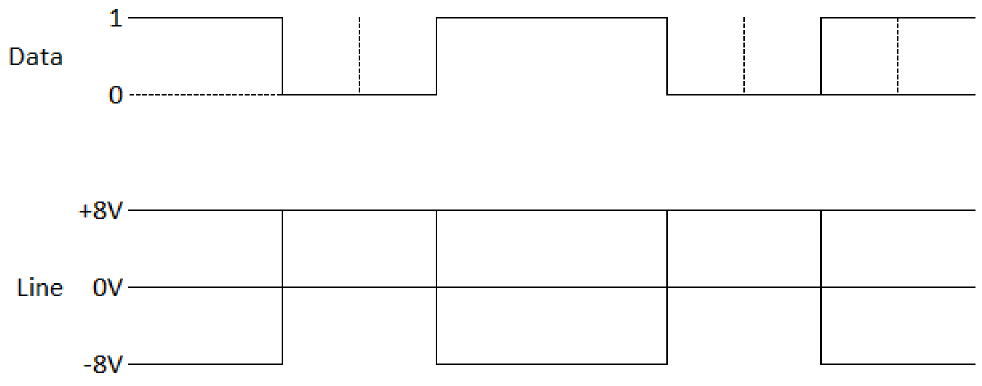


Figure 4.3: Relationship between data and signal states.

The status signal lines are `RTS` (output), `CTS` (input), `DTR` (output), `DSR` (input) and `CD` (input). These signals are ON when high and OFF when low (Figure 4.4).

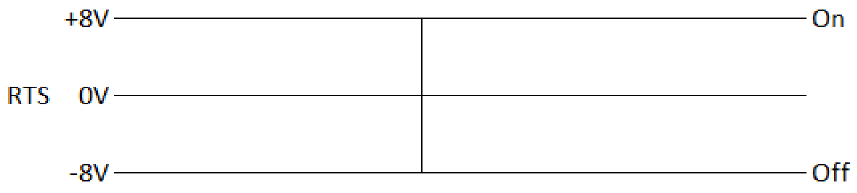


Figure 4.4: Signal line output status (`RTS`)

The HX-20 is provided with two types of interfaces. These are serial and RS-232C. The serial interface uses the serial port of the MCU and the bit rates and word length are fixed. The RS-232C interface, however, performs handshaking by software. It can therefore support bit rates up to 4800bps and both the bit rate and word length can be set by the user. Table 4.1 shows the respective range of functions for the serial and RS-232C interfaces.

The serial interface is used for communication between the master and slave MCUs and for the floppy disk units.

	Transmission speed	Word length (bits)	Stop bit (bits)	Control lines (input)	Control lines (output)
Serial	38.4Kbps 4.8Kbps 600bps 150bps	8		1	1
RS232C	4.8Kbps max.	5, 6, 7 or 8	1 or 2	3 (`CTS`, `DSR` and `CD`)	2 (`RTS` and `DTR`)

Table 4.1: Functions of serial and RS-232C interfaces

4.2 I/O ports

MCU	Port (address)	Input / output	Signal name or function	Signal state	Port bit state
Master	`P10`	Input	`DSR` (RS-232C)	High	0
				Low	1
	`P11`	Input	`CTS` (RS-232C)	High	0
				Low	1
	`P16`	Input	`PIN` (serial control line)	High	0
				Low	1
	`P21`	Output	Selection of slave or serial for CPU serial communication		0: Slave 1: Serial
	`RMCR` (`0010`)	Output	Serial bit rate control		
	`TRCSR` (`0011`)	Input	Serial control and status signal		
	`SRDR` (`0012`)	Input	Serial receive data		
	`STDR` (`0013`)	Output	Serial transmit data		
	`$26` bit 5	Output	`POUT` (serial control line)	High	0
				Low	1

Continues in next page...

…continued from previous page

MCU	Port (address)	Input / output	Signal name or function	Signal state	Port bit state
Slave	P20	Input	RXD (RS-232C)	High	0
				Low	1
	P31	Output	RTS (RS-232C)	High	0
				Low	1
	P36	Output	Serial and RS-232C interface driver ON / OFF		0: ON ; 1: OFF
	P45	Output	P20 signal selection		0: RS232 1: micro-cassette
	P47	Input	CD (RS-232C)	High	0
				Low	1

Table 4.2: I/O ports for serial communication

Note:

- `DSR`: data set ready
- `CTS`: clear to send
- `TXD`: transmit data
- `RTS`: request to send
- `DTR`: data terminal ready
- `RXD`: receive data
- `CD` : carrier detect

4.3 Serial communication procedure

The SCI (serial communication interface) in the MCU performs serial communication in the following procedure.

1. Driver ON

   The communication driver is turned ON. The port for driver ON is connected to the slave MCU. Subroutine `SERONF` turns the drivers ON/OFF.

2. Serial switching

   The serial communication lines of the MCU can be used either for external data communication or for communication with the slave MCU. Normally, the slave MCU is selected. To select external communication, port `P22` of the main MCU is set to `1`.

3. Bit rate setting

   `RMCR` (address `10`) sets the bit rate. The bit rate is normally set to 38.4Kbps. Table 4.3 shows selection of bit rates by `RMCR`.

Lower 4 bits	Hexadecimal	Bit time / bit rate
`0100`	`4`	26µs / 38.4Kbps
`0101`	`5`	208µs / 4.8Kbps
`0110`	`6`	1.67ms / 600bps
`0111`	`7`	6.67ms / 150bps

Table 4.3: Bit time and bit rates

4. Data transmission (one byte)

   `TRCSR` (address `0011`) is input and when it is confirmed that `TDRE` (bit 5 of `TRCSR`) is `1`, one byte of data is transmitted by writing it to `TDR` (address `0013`).

5. Data reception (one byte)

   `TRCSR` is input and if `RDRF` (bit 7 of `TRCSR`) is `1`, serial data can be received by `SRDR` (address `0012`). One byte of serial data is then received. Note that if the received data is not fetched before the next data is received, an overrun error occurs (`ORFE` is set at `1`).

6. Termination procedure

   The bit rate is set to 38.4Kbps (see step 3 above) and the driver is turned OFF. The procedure is followed because transmission of commands to the slave MCU is always performed at 38.4Kbps.

4.4 Control lines

Two control lines are available: `PIN` (input) and `POUT` (output). `PIN` is connected to `P16` (bit 6 of port 1) and `POUT` is connected to bit 5 of address `26`. Both of these signals are set at `1` when the signal goes high and at `0` when the signal goes low. Subroutine `WRTP26` is used to set data in address `26`. Figure 4.5 shows the relationship of the signals and ports.

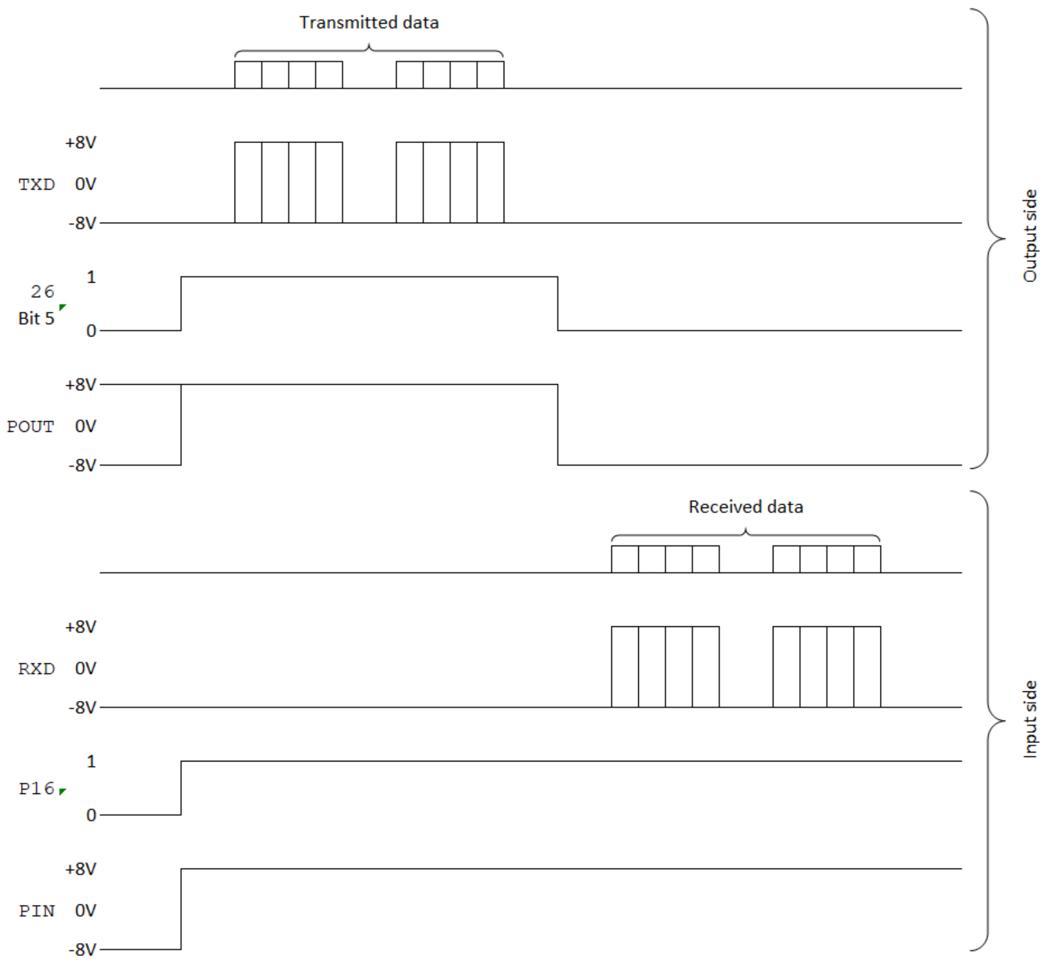


Figure 4.5: Relationship of signals and ports

4.5 High-speed serial communication

EPSP (EPSON Serial Communication Protocol) is provided to enable serial communication between the HX-20 and a floppy disk unit (TF-20) or between two HX-20s.

Figure 4.6 shows how slave devices can be connected by data lines to the HX-20. Up to two slave devices can be connected to a single master device. Each slave is assigned a device number by the master. The master then uses the device number to select which of the slaves to perform communication with. The master can only communicate with one slave at a time. Communication between slave devices cannot be performed.

Figure 4.7 shows the format for messages sent from the HX-20 to a slave

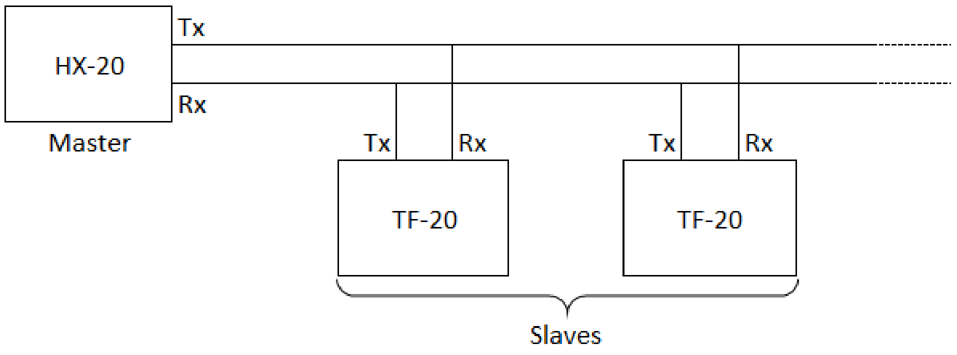


Figure 4.6: Connection of slave devices to HX-20 for serial communication

device.

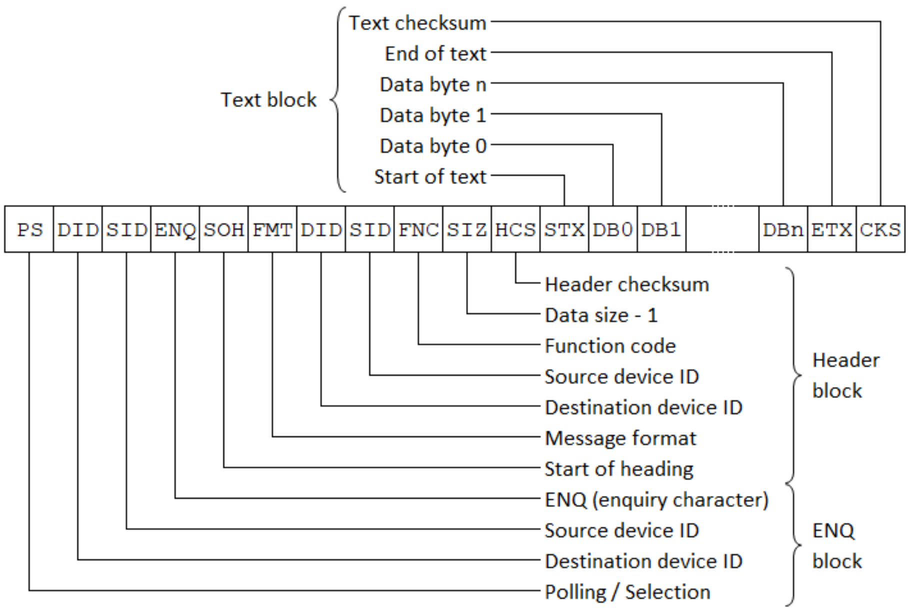


Figure 4.7: Message format

Messages sent from the master can be divided into three blocks described below.

1. `ENQ` block

   `PS` to `ENQ` in Figure 4.7. The master sends this block to request connection with a slave.

2. Header block

`SOH` to `HCS` in Figure 4.7. This block specifies the data format etc.

3. Text block

   `STX` to `CKS` in Figure 4.7. The text block contains the actual data transmitted.

Details of each block are as follows

4.5.1 `ENQ` block

The contents of the `ENQ` block are shown below.

PS	DID	SID	ENQ

The master device selects one of the slave devices and issues a connection request to it, using this block. When connection with the slave has been established, the header and text blocks are sent. Once a slave device has been connected, this procedure is not repeated until a new slave device is selected for communication. The selected slave device issues an `ACK` signal in response to the connection request from the master (Figure 4.8).

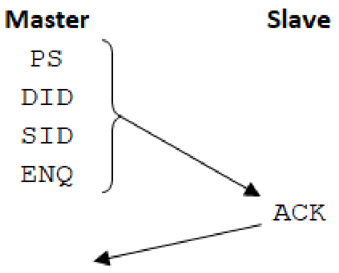


Figure 4.8: `ENQ` block procedure

- `PS` specifies polling/selection. At present, however, only selection is supported. The code for `PS` is $31_{16} = 1$.
- `DID` indicates the destination device ID. The following destination device IDs are available:
  - 31_{16}: floppy disk drive `A`.
  - 32_{16}: floppy disk drive `B`.
  - 33_{16}: floppy disk drive `C`.
  - 34_{16}: floppy disk drive `D`.
  - 20_{16}: master HX-20.
- The code for `ENQ` is 05_{16}.

4.5.2 Header block

The master transmits the header block to specify the message format and the function codes as well as text size to be sent to the floppy disk unit in the text block that follows.

The contents of the header block are shown below.

SOH	FMT	DID	SID	FNC	SIZ	HCS

- `SOH` indicates the start of the header. The value is `01`.
- `FMT` indicates the block format.
  - `00` indicates that the master device is transmitting a message to a slave device.
  - `01` indicates that a slave device is transmitting a message to the master device.
- `DID` indicates the destination device ID. The codes for `DID` are the same as in the `ENQ` block.
- `SID` indicates the source device ID.
- `FNC` specifies the function of the disk unit. Must be `00` to `FF`. For details of each function, refer to the descriptions in the corresponding sections.
- `SIZ` indicates the text block size. This value is the number of bytes in the text block (excluding `STX` and `CKS`) minus 1. The value of `SIZ` must be in the range 0 to 255_{10}.
- `HCS` indicates the checksum of the header block. The value is such that the lower 8 bits of the sum of the values of the header block (`SOH` to `HCS`) will all be 0.

When the slave device receives a correct header block, it responds by sending '`ACK`' to the corresponding source device. If the slave device receives an incorrect header block, it responds by sending '`NAK`' to the source device.

4.5.3 Text block

The text block contains the actual data to be sent to the selected device. The text block follows the header block.

The contents of the text block are shown below.

STX	$\mathtt{DB}_0$	$\mathtt{DB}_1$	...	$\mathtt{DB}_n$	ETX	CKS

- `STX` indicates the start of the text. The value is `02`.
- $\texttt{DB}_0$: data 0.
- $\texttt{DB}_n$: data n ($n \leq 255$).
- `ETX` indicates the end of the text.
- `CKS` indicates the checksum of the text block. The value is such that the lower bits of the sum of the values of the text block (`STX` to `CKS`) will be 0.

When the slave device receives a correct text block, it responds by sending `ACK` to the source device. If the slave device receives an incorrect text block, it responds by sending '`NAK`' to the source device.

4.5.4 Switching transmit state

There are cases when the master (HX-20) will request data transmission from a slave device (e.g., floppy disk unit). In this case, the sending and receiving sides (master and slave) are reversed. Switching over from master-to-slave to slave-to-master data transmission is accomplished by the following procedure.

The master sends `EOT` (code `04`) to the slave after it has received an `ACK` from the slave indicating the slave has correctly received the text block. The slave device, after receiving `EOT`, sends the header and text blocks to the master device. It then sends `EOT` to the master and the transmit state returns to master-to-slave (Figure 4.9).

Details of the protocol are shown in the EPSP standard at Appendix A.

4.6 Subroutines for serial communication

The following four subroutines support serial communication using EPSP procedures:

1. `SERONF`: turns ON/OFF the serial communication drivers.
2. `SEROUT`: transmits the `ENQ`, header and text blocks.
3. `SERIN`: receives the header and text blocks.
4. `SRINIT`: sets constants and performs initialization.

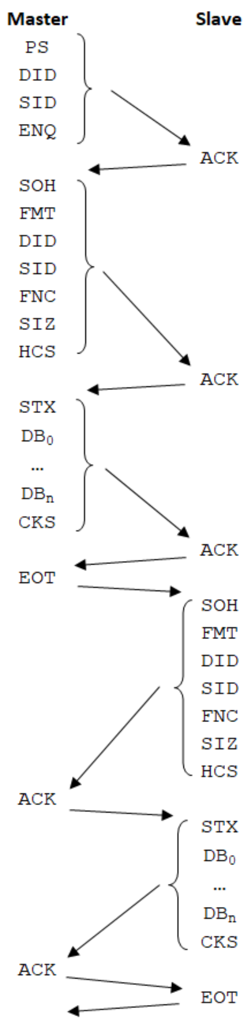


Figure 4.9: Transmit state switch

4.7 High-speed serial subroutines

Subroutine name	Entry point	Description
SERONF	FF73	Turns ON/OFF the high-speed serial driver. This subroutine checks bit 4 of 'SRSTS' and turns ON the driver only when both are off.
		The contents of the SERONF parameters are the same as those of RSONOF.
SEROUT	FF70	High-speed serial data output (EPSP-based data transmission). This subroutine transmits the ENQ, header and text blocks to the specified device according to the ENQ...SOH...ETX procedure.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters – At entry ∗ `(X)`: head address of a data packet ∗ `(A)`: indicates whether to proceed to the receive procedure after completion of the transmit procedure. · `00`: transmit procedure only. · `01`: (LSB=1) proceeds to the receive procedure after completion of the transmit procedure. Packets 1. `FMT` (1 byte) 2. `DID` (1 byte) 3. `SID` (1 byte) 4. `FNC` (1 byte) 5. `SIZ` (1 byte) 6. (data string) (1 byte) ... *n* – At return ∗ `(C)`: abnormal I/O flag ∗ `(A)`: return codes · `00`: normal end · `B0`: time out · `B1`: not linked (device error) · `B2`: communication error · `B3`: driver OFF. ∗ `(Z)`: according to the value of `(A)` • Registers retained: none • Subroutines referenced: `CHKRS` • Variables used: `R0`, `R1`, `R2`, `R3`, `R4` and `R5H`

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
SERIN	FF6D	Receives the header and text blocks according to the SOH...STX procedure (high-speed serial data block reception).
		• Parameters – At entry ∗ (X): head address of receive data block. – At return ∗ (C): abnormal I/O flag ∗ (A): return codes · 00: normal · B0: time out · B2: error during receive procedure ∗ (B): indicates the receive block status when (A) is 00 · 00: data with a header string (SOH...) received · 01: data without a header string received ∗ (Z): according to the value of (A) Note: the format of a data block received is the same as that of the data block transmitted (see SEROUT subroutine). • Registers retained: none • Subroutines referenced: CHKRS • Variables used: R0, R1, R2, R3, R4 and R5H

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
`SRINIT`	`FF7C`	Sets constants and performs high-speed serial initialization. Values of constants on initialization: • `SRTCN` $\leftarrow 3$ • `SRTMO` $\leftarrow 10_{10}$ • `SREMO` $\leftarrow 100_{10}$ • `SRAMO` $\leftarrow 10_{10}$ • `SRTDL` $\leftarrow 1$ • Others $\leftarrow 0$
		• Parameters – At entry ∗ `(A)`: value of `SRMODE` `00` or `01` (`00`: master) – At return: none • Registers retained: none • Subroutines referenced: none • variables used: none

Table 4.4: High-speed serial subroutines

4.8 High-speed serial communication work areas

Address (from)	Address (to)	Variable name	Bytes	Description
1C4	1C4	SRFMT	1	FMT (format) data
1C5	1C5	SRDDEV	1	DID (destination device ID) data
1C6	1C6	SRSDEV	1	SID (source device ID) data
1C7	1C7	SRFNC	1	FNC (function) data
1C8	1C8	SRSIZ	1	SIZ (size) data
1C9	1C9	SRACKC	1	ACK character (sent from destination device on completion of block transmission)
1CA	1CA	SRTRCN	1	Number of times same block has been sent
1CB	1CB	SRTIMO	1	Time out for received characters (unit: ms)
1CC	1CC	SRETMO	1	Time out for received block reception (unit: ms)
1CD	1CD	SRATMO	1	Time out for received ACK characters (unit: ms)
1CE	1CE	SRMODE	1	Relationship between devices (0: master; any other value: slave)
1CF	1CF	SRETDL	1	Idle time after EOT transmission (unit: ms)
1D0	1D0	SRBLCN	1	Number of received data (block reception)
1D1	1D1	SRERMD	1	Error (block reception)
1D2	1D2	SRRVFL	1	Not used
1D3	1D4	SREIX	2	Address where received data is stored (block reception)

Table 4.5: High-speed serial communication work areas

4.9 Sample listings: serial communication between two HX-20

```
0 :                        ; EPSP
0 :                        ; Read character from keyboard and send
0 :                        ; characters to another HC-20 by EPSP.
0 :                        ; And at another HC-20.
0 :                        ; Received character from EPSP, display
0 :                        ; characters on the virtual screen.
0 :                        ;
0 :                        ; Serial communication
0 :                             PAGE   0
0 :                             CPU    6301
0 :                        ;
0 :                        ; Condition switch
0 : =$0                    SRSL   EQU    0      ; Select serial procedure
0 :                        ;
0 :                        ; Common definitions
0 :                        ;
0 :                        ; MPU 6301 I/O ports
0 : =$2                    PORT1  EQU    $02    ; I/O port 1
0 : =$3                    PORT2  EQU    $03    ; I/O port 2
0 : =$6                    PORT3  EQU    $06    ; I/O port 3
0 :                        ;
0 :                        ; Other registers
0 : =$11                   TRCSR  EQU    $11    ; Transmit/receive control & status registers
0 : =$10                   RMCR   EQU    $10    ; Rate & mode control register
0 : =$13                   STDR   EQU    $13    ; Serial transmit data register
0 : =$12                   SRDR   EQU    $12    ; Serial data receive data register
0 :                        ;
0 :                        ; Subroutine entry point
0 : =$FF4F                 DSPSCR EQU    $FF4F  ; Display one character to virtual screen
0 : =$FF5E                 SCRFNC EQU    $FF5E  ; Virtual screen function
0 : =$FF6D                 SERIN  EQU    $FF6D  ; Serial receive
0 : =$FF70                 SEROUT EQU    $FF70  ; Serial transmit
```

```
33/       0 : =$FF73                SERONF EQU    $FF73  ; Serial driver on/off
34/       0 : =$FF9A                KEYIN  EQU    $FF9A  ; Get one character from keyboard buffer
35/       0 : =$FF9D                KEYSTS EQU    $FF9D  ; Get number of characters in the key buffer
36/       0 :                       ;
37/      50 :                              ORG    $50
38/      50 :                       ; General registers used by I/O routine
39/      50 :                       R0H    RMB    1
40/      51 :                       R0L    RMB    1
41/      52 : =$50                  R0     EQU    R0H    ; 2 bytes register (R0H,R0L)
42/      52 :                       R1H    RMB    1
43/      53 :                       R1L    RMB    1
44/      54 : =$52                  R1     EQU    R1H    ; 2 bytes register (R1H,R1L)
45/      54 :                       R2H    RMB    1
46/      55 :                       R2L    RMB    1
47/      56 : =$54                  R2     EQU    R2H    ; 2 bytes register (R2H,R2L)
48/      56 :                       R3H    RMB    1
49/      57 :                       R3L    RMB    1
50/      58 : =$56                  R3     EQU    R3H    ; 2 bytes register (R3H,R3L)
51/      58 :                       R4H    RMB    1
52/      59 :                       R4L    RMB    1
53/      5A : =$58                  R4     EQU    R4H    ; 2 bytes register (R4H,R4L)
54/      5A :                       R5H    RMB    1
55/      5B :                       R5L    RMB    1
56/      5C : =$5A                  R5     EQU    R5H    ; 2 bytes register (R5H,R5L)
57/      5C :                       ;
58/      7A :                              ORG    $7A
59/      7A :                       SRSTS  RMB    1      ; Serial status
60/      7B :                                            ; Bit 0,1: RS232 mode (00:Stop 01:Interrupr read
61/      7B :                                            ;                      02:Read one character)
62/      7B :                                            ; Bit 2: Execute/pause (0:On execute 1:Pause)
63/      7B :                                            ; Bit 3: RS232 driver (0:Off 1:Driver on)
64/      7B :                                            ; Bit 4: Serial driver (0:Off 1:Driver on)
```

```
 7B :                                  ; Bit 5,6,7: CPU serial receive interrupt mode
 7B :                   RUNOD  RMB    1      ; Run mode ($80:BASIC $00:System)
 7C :                   SIOSTS RMB    1      ; Slave I/O status (each bit 0:Off 1:On)
 7D :                                  ; Bit 0: Printer
 7D :                                  ; Bit 1: External cassette
 7D :                                  ; Bit 2: Internal cassette
 7D :                                  ; Bit 3: RS232 on (read)
 7D :                                  ; Bit 4: Speaker on
 7D :                                  ; Bit 5: PROM cassette
 7D :                                  ; Bit 6: Barcode reader
 7D :                                  ; Bit 7: Break slave CPU (0: on execute
 7D :                                  ;                       1: broken by interrupt)
 7D :                   MIOSTS RMB    1      ; Main I/O status (each bit 0:off 1:on)
 7E :                                  ; Bit 0: LCD on read/write characters
 7E :                                  ; Bit 1: on continue send command to slave CPU
 7E :                                  ; Bit 2: on continue to send serial line
 7E :                                  ; Bit 3: on clock interrupt
 7E :                                  ; Bit 4: (power fail)
 7E :                                  ; Bit 5: (off power switch)
 7E :                                  ; Bit 6: on pause key
 7E :                                  ; Bit 7: on break key
 7E :                   ; RAM common area
1C4 :                          ORG    $1C4
1C4 :                   ; Work for serial communication
1C4 :                   SRFMT  RMB    1      ; Format                             (0)
1C5 :                   SRDDEV RMB    1      ; Destination device                 (1)
1C6 :                   SRSDEV RMB    1      ; Source device                      (2)
1C7 :                   SRFNC  RMB    1      ; Function                           (3)
1C8 :                   SRSIZ  RMB    1      ; Text size                          (4)
1C9 :                   SRACKC RMB    1      ; Received ACK character             (5)
1CA :                   SRTRCN RMB    1      ; Send try count                     (6)
1CB :                   SRTIMO RMB    1      ; For receive character time over limit   (7)
```

```
 97/     1CC :                    SRETMO RMB    1      ; For receive block time over limit       (8)
 98/     1CD :                    SRATMO RMB    1      ; For receive ACK time over limit         (9)
 99/     1CE :                    SRMODE RMB    1      ; Serial master/slave mode (0:master)     (10)
100/     1CF :                                         ;                           (NOT:slave)
101/     1CF :                    SRETDL RMB    1      ; After send EOT, idling time (1 = 1 ms)  (11)
102/     1D0 :                    SRBLCN RMB    1      ; For receive block, block counter
103/     1D1 :                    SRERMD RMB    1      ; For receive block, error mode
104/     1D2 :                    SRRVFL RMB    1      ; For receive block, received character flag
105/     1D3 :                    SREIX  RMB    2      ; For receive block, data stored address
106/     1D5 : =$1C4              SRWKTP EQU    SRFMT  ; Serial work top address
107/     1D5 : =$1D5              SRWKBT EQU    SREIX+2; Work bottom
108/     1D5 :                    ;
109/     1D5 :                    ; Serial communication routine
110/     1D5 :                    ;
111/     1D5 : =$1                SOH    EQU    $01    ; SOH
112/     1D5 : =$2                STX    EQU    $02    ; STX
113/     1D5 : =$3                ETX    EQU    $03    ; ETX
114/     1D5 : =$4                EOT    EQU    $04    ; EOT
115/     1D5 : =$5                ENQ    EQU    $05    ; ENQ
116/     1D5 : =$6                ACK    EQU    $06    ; ACK
117/     1D5 : =$15               NAK    EQU    $15    ; NAK
118/     1D5 : =$10               DLE    EQU    $10    ; DLE
119/     1D5 : =$38               WAK    EQU    $38    ; WAK (;)
120/     1D5 : =$31               DEVCRT EQU    $31    ; Device no. (CRT)
121/     1D5 :                    ;
122/    1000 :                           ORG    $1000
123/    1000 :                    ;
124/    1000 :                    ; Out to serial routine
125/    1000 :                    ;
126/    1000 :                    ; Inicialization of serial
127/    1000 :                    ; 1. Clear FMT, DID, SID, FNC, SIZ work
128/    1000 :                    ; 2. Set NAK code to ACK character area
```

```
1000 :                  ; 3. Set retry count (initial 5)
1000 :                  ; 4. Set time over count (initial 0.5 sec)
1000 :                  ; 5. Set start block time over count (initial 10 sec)
1000 :                  ;
1000 :                  ; Receive one character from serial port
1000 :                  ; On entry
1000 :                  ;   Parameter none
1000 :                  ; On exit
1000 :                  ;   (C): I/O error flag 0:OK 1:error
1000 :                  ;   (V): Time over flag 0:OK 1:time over (time over = 0.1 sec)
1000 :                  ;   (A): received character (if (C)=0 and (Z)=1)
1000 :                  ; Register preserve B,X
1000 :                  ;
1000 : B6 01 CB         SRVSGL LDAA   SRTIMO ; Set time over counter
1003 :                  ; Entry point (parameter (A): time over limit)
1003 : 37               SRVSXX PSHB
1004 : 16                      TAB
1005 : 8D 0D            SRVS40 BSR    SRVBYT ; Receive one character
1007 : 25 09                   BCS    SRVS50 ; I/O error?
1009 : 28 07                   BVC    SRVS50 ; OK?
100B : 5D                      TSTB          ; Time over limit check
100C : 27 F7                   BEQ    SRVS40
100E : 5A                      DECB
100F : 26 F4                   BNE    SRVS40
1011 : 0B                      SEV           ; Time out
1012 : 33               SRVS50 PULB
1013 : 39                      RTS
1014 :                  ;
1014 :                  ; Receive one bytes
1014 :                  ; Register preserve B,X
1014 :                  ;
1014 : 3C               SRVBYT PSHX
```

```
1015 : CE 0B B8               LDX    #3000         ; 21 * 1.6 * 3000 = 100,000
1018 : 4F              SRVS10 CLRA                 ; (C) <- 0, (V) <- 0           (1 C/S)
1019 : 09                     DEX                  ; Not received, check time over (1 C/S)
101A : 0B                     SEV                  ; Preset (V)                   (1 C/S)
101B : 27 12                  BEQ    SRVS30        ;                              (2 C/S)
101D : 0D                     SEC                  ; Preset I/O error flag        (1 C/S)
101E : 7B B0 7D               TIM    #$B0,MIOSTS   ;                              (3 C/S)
1021 : 26 0C                  BNE    SRVS30        ;                              (2 C/S)
1023 : 7B 04 03               TIM    #$4,PORT2     ; Connected external serial    (3 C/S)
1026 : 26 07                  BNE    SRVS30        ;                              (2 C/S)
1028 : 7D 00 11               TST    TRCSR         ; Received?                    (3 C/S)
102B : 2A EB                  BPL    SRVS10        ;                              (2 C/S) 21 C/S
102D :                 ; Received                  ; (A) <- Received character
102D : 96 12                  LDAA   SRDR          ; (C),(V) <- 0 by TST instruction
102F : 38              SRVS30 PULX
1030 : 39                     RTS
1031 :                 ; Wait to be selected
1031 :                 ;   Receive sequence
1031 :                 ;   1. Wait serial idling
1031 :                 ;   2. Check EOT
1031 :                 ; Parameter
1031 :                 ; On entry
1031 :                 ;   (A): destination device (for sending side)
1031 :                 ;   (B): source device      (for sending side)
1031 :                 ;   (X): time over limit (1=0.1 sec, 0:no limit)
1031 :                 ; On exit
1031 :                 ;   (C): I/O error flag (0:normal 1:error)
1031 :                 ;   (A): return code (0:normal) ($B3: time out error)
1031 :                 ;   (Z): depend on value of (A)
1031 :                 ;
1031 :                 ; Work use as register
1031 :                 ;   R0H:31
```

```
1031 :                  ;   R0L:DID
1031 :                  ;   R1H:SID
1031 :                  ;   R1L:ENQ
1031 :                  ;
1031 :                  ; Entry point: received EOT, check ENQ pattern
1031 : DD 51            SRSLET STD    R0L    ; Set ENQ pattern
1033 : 86 05                   LDAA   #ENQ
1035 : 97 53                   STAA   R1L
1037 : 86 31                   LDAA   #$31
1039 : 97 50                   STAA   R0H    ; Set P1
103B : 20 2E                   BRA    SRSL18
103D :                  ;
103D :                  ; Entry point: wait EOT P1 ... ENQ pattern
103D : DD 51            SRSLCT STD    R0L    ; Set ENQ pattern
103F : 86 05                   LDAA   #ENQ
1041 : 97 53                   STAA   R1L
1043 : 86 31                   LDAA   #$31
1045 : 97 50                   STAA   R0H    ; Set P1
1047 :                  ;
1047 : 71 FB 03         SRSL10 AIM    #$FF-$4,PORT2 ; Select external serial
104A : 72 01 11         SRSL11 OIM    #$1,TRCSR     ; Set wake up flag
104D : 7B 01 11         SRSL13 TIM    #$1,TRCSR     ; Idle?
1050 : 27 0E                   BEQ    SRSL15
1052 : 0D                      SEC
1053 : 7B 04 03                TIM    #$4,PORT2
1056 : 26 30                   BNE    SRSL30        ; Broken serial?
1058 : 96 11                   LDAA   TRCSR         ; Ignore received character
105A : 2A F1                   BPL    SRSL13
105C : 96 12                   LDAA   SRDR
105E : 20 EA                   BRA    SRSL11
1060 : BD 10 14         SRSL15 JSR    SRVBYT        ; Read character
1063 : 25 23                   BCS    SRSL30        ; I/O error?
```

```
1065 : 29 22                  BVS    SRSL40
1067 :                 ; Received EOT?
1067 : 81 04                  CMPA   #EOT
1069 : 26 1E                  BNE    SRSL40
106B : 5F              SRSL18 CLRB                  ; (B):received counter DID:0 SID:1
106C : BD 10 14        SRSL20 JSR    SRVBYT
106F : 29 18                  BVS    SRSL40         ; Time over?
1071 : 25 15                  BCS    SRSL30         ; I/O error?
1073 : 3C                     PSHX
1074 : CE 00 50               LDX    #R0
1077 : 3A                     ABX
1078 : A1 00                  CMPA   0,X
107A : 38                     PULX
107B : 26 0C                  BNE    SRSL40
107D : 5C                     INCB
107E : C1 04                  CMPB   #4             ; Received 00 DID SID ENQ ?
1080 : 26 EA                  BNE    SRSL20
1082 :                 ; Received ACK sequence
1082 : 86 06                  LDAA   #ACK
1084 : BD 10 98               JSR    SSRSGL
1087 : 4F                     CLRA
1088 : 39              SRSL30 RTS
1089 :                 ; Time out?
1089 : 8C 00 00        SRSL40 CPX    #0             ; Check time out?
108C : 27 B9                  BEQ    SRSL10
108E : 09                     DEX
108F : 26 B6                  BNE    SRSL10
1091 : 86 B3                  LDAA   #$B3           ; Time out error return
1093 : 7F 01 C5               CLR    SRDDEV
1096 : 4D                     TSTA
1097 : 39                     RTS
1098 :                 ;
```

```
1098 :                   ; Send one character to serial port
1098 :                   ; On entry
1098 :                   ;   (A): send character
1098 :                   ; On exit
1098 :                   ;   (C): I/O break flag 0:OK 1:error
1098 :                   ; Register preserve all
1098 :                   ;
1098 : 0D                SSRSGL SEC                  ; Preset I/O error flag
1099 : 7B 80 7D                 TIM    #$80,MIOSTS   ; Break?
109C : 26 0D                    BNE    SSRS20
109E : 7B 04 03                 TIM    #$4,PORT2     ; Connect to external serial?
10A1 : 26 08                    BNE    SSRS20
10A3 : 7B 20 11                 TIM    #$20,TRCSR    ; Send ready?
10A6 : 26 F0                    BNE    SSRSGL
10A8 : 97 13                    STAA   STDR
10AA : 0C                       CLC                  ; OK return
10AB : 39                SSRS20 RTS
10AC :                   ; Serial initialize
10AC :                   ; On entry
10AC :                   ;   (A): mode (master:0 slave:nonzero)
10AC :                   ;
10AC : 36                SRINIT PSHA
10AD : CC 00 11                 LDD    #SRWKBT-SRWKTP; Clear work (A):pattern, (B):count
10B0 : CE 01 C4                 LDX    #SRWKTP
10B3 : 3C                       PSHX
10B4 : A7 00             CLEARB STAA   0,X
10B6 : 08                       INX
10B7 : 5A                       DECB
10B8 : 26 FA                    BNE    CLEARB
10BA :                   ;
10BA : 38                       PULX
10BB : 32                       PULA
```

```
10BC : A7 0A                      STAA   SRMODE-SRWKTP,X ; Set master/slave mode
10BE : 62 03 06                   OIM    #3,SRTRCN-SRWKTP,X ; Set retry count = 3
10C1 : CC 0A 64                   LDD    #10*256+100
10C4 : ED 07                      STD    SRTIMO-SRWKTP,X ; Time over limit = 1 sec
10C6 :                                                   ; Receive block time over limit=10
10C6 : A7 09                      STAA   SRATMO-SRWKTP,X ; Receive ACK time over = 1 sec
10C8 : 6C 0B                      INC    SRETDL-SRWKTP,X ; After EOT, idling time
10CA :                     ;
10CA : 39                         RTS
10CB :                     ;
10CB :                     ; Receive from serial (for slave device)
10CB :                     ; On entry
10CB :                     ;   (X): received data stored address
10CB :                     ; On exit
10CB :                     ;   (A): return code   0:OK $B0:time over $B2:receive error
10CB :                     ;                      $B8:received EOT
10CB :                     ;   (B): 0:received with header 1:received without header
10CB :                     ;        (effective (A)=0)
10CB :                     ;   (Z): depend on value of (A)
10CB :                     ;
10CB :                     ; Work use as register
10CB :                     ;   R0H: top character of block ($1 or $2)
10CB :                     ;   R1 : address of stored data
10CB :                     ;   R2L: block length
10CB :                     ;   R3H: retry count
10CB :                     ;   R3L: time over counter
10CB :                     ;   R4H: top character of block
10CB :                     ;   R4L: omitted header flag (0:not 1:omitted)
10CB :                     ;
10CB :                     ; Error return routine
10CB :                     ;   Set error code to (A), clear (C)
10CB :                     ;
```

```
10CB : 86 B0             SRERB0 LDAA   #$B0   ; Error $B0 (time over)
10CD : 20 0A                    BRA    SRER10
10CF :                   ;
10CF : 86 B1             SRERB1 LDAA   #$B1   ; Error $B1
10D1 : 20 06                    BRA    SRER10
10D3 : 86 B8             SRERB8 LDAA   #$B8   ; Error $B8 (received EOT)
10D5 : 20 02                    BRA    SRER10
10D7 :                   ;
10D7 : 86 B3             SRERB3 LDAA   #$B3   ; Error $B3 (driver off)
10D9 : 7F 01 C5          SRER10 CLR    SRDDEV ; Clear DID (for start from end process)
10DC : 7E 11 7F          SRER20 JMP    SRRB90
10DF :                   ;
10DF :                   ; EPSP receive (slave device) subroutine
10DF :                   ; Receive from serial
10DF :                   ; On entry
10DF :                   ;   (X): received data stored address
10DF :                   ;
10DF : DF 52             SERRCV STX    R1
10E1 : 7B 10 7A                 TIM    #$10,SRSTS    ; Driver on?
10E4 : 27 F1                    BEQ    SRERB3
10E6 :                   ;
10E6 :                   ; Select serial (detach slave)
10E6 : 96 11                    LDAA   TRCSR         ; Save TRCSR for recover RS232
10E8 : 97 5B                    STAA   R5L
10EA :                   ;
10EA : 0F                SERINS SEI
10EB : 71 EF 11                 AIM    #$FF-$10,TRCSR ; Serial interrupt disable
10EE : 71 FB 03                 AIM    #$FF-$4,PORT2
10F1 : 4F                INSR05 CLRA
10F2 : 4C                       INCA
10F3 : 97 59             SRRB10 STAA   R4L           ; Omitted header block (initial)
10F5 : B6 01 CA          SRRB20 LDAA   SRTRCN        ; Set retry count
```

```
10F8 : 97 56                  STAA   R3H
10FA :                 ; Receive first character
10FA : DE 52           SRRB30 LDX    R1            ; (X): stored data address
10FC : B6 01 CC               LDAA   SRETMO        ; Set time over for waiting block
10FF : BD 10 03               JSR    SRVSXX
1102 : 25 D8                  BCS    SRER20
1104 : 29 C5                  BVS    SRERB0        ; Time over error?
1106 : C6 04                  LDAB   #4            ; (B): block size (preset for header block)
1108 : 81 01                  CMPA   #SOH
110A : 27 2E                  BEQ    SRRB50
110C : 5C                     INCB                 ; (B): 5
110D : 3A                     ABX                  ; (X): data stored address
110E : F6 01 C8               LDAB   SRSIZ         ; (B): block size (for data block)
1111 : 81 02                  CMPA   #STX
1113 : 27 25                  BEQ    SRRB50
1115 : 81 05                  CMPA   #ENQ
1117 : 27 19                  BEQ    SRCE10
1119 : 81 04                  CMPA   #EOT          ; EOT?
111B : 27 B6                  BEQ    SRERB8
111D :                 ; Other codes (skip current block and send NAK)
111D : BD 10 14        SRRB40 JSR    SRVBYT        ; Received one character?
1120 : 25 5D                  BCS    SRRB90
1122 : 28 F9                  BVC    SRRB40
1124 :                 ; Time over (not received data 0.1 sec)
1124 :                 ;
1124 :                 ; Error NAK send
1124 : 86 15           SRCSER LDAA   #NAK
1126 : B7 01 C9               STAA   SRACKC        ; Set NAK character for ENQ
1129 : D6 56                  LDAB   R3H           ; Retry count check
112B : 27 A2                  BEQ    SRERB1
112D : 7A 00 56               DEC    R3H
1130 : 27 9D                  BEQ    SRERB1
```

```
1132 :                ; Entry from ENQ
1132 : B6 01 C9       SRCE10 LDAA   SRACKC
1135 : BD 10 98              JSR    SSRSGL         ; Send NAK
1138 : 20 C0                 BRA    SRRB30
113A :                ;
113A :                ; Receive data block (SOH... or STX...)
113A :                ;
113A : 5C             SRRB50 INCB
113B : D7 55                 STAB   R2L
113D : 97 58                 STAA   R4H            ; R4H: received first character
113F : 97 50                 STAA   R0H            ; R0H: $1 (SOH) or $2 (STX)
1141 : 16                    TAB                   ; (B): checksum
1142 :                ; Receive data string loop
1142 : BD 10 00       SRRB70 JSR    SRVSGL
1145 : 25 38                 BCS    SRRB90
1147 : 29 DB                 BVS    SRCSER         ; Time over?
1149 : A7 00                 STAA   0,X
114B : 08                    INX
114C : 1B                    ABA
114D : 16                    TAB
114E : 7A 00 55              DEC    R2L
1151 : 26 EF                 BNE    SRRB70
1153 :                ;
1153 : BD 10 00       SRRB75 JSR    SRVSGL         ; Receive checksum
1156 : 25 27                 BCS    SRRB90
1158 : 29 CA                 BVS    SRCSER
115A : 1B                    ABA
115B : 16                    TAB
115C : 7A 00 50              DEC    R0H            ; If STX... receive ETX
115F : 26 F2                 BNE    SRRB75
1161 : 5D                    TSTB                  ; Checksum OK?
1162 : 26 C0                 BNE    SRCSER
```

```
1164 :                  ;
1164 : 86 06                  LDAA   #ACK
1166 : B7 01 C9               STAA   SRACKC        ; Save send ACK code for ENQ
1169 : BD 10 98               JSR    SSRSGL
116C : 25 11                  BCS    SRRB90
116E : DC 58                  LDD    R4            ; R4H <- First character of block,
1170 :                                              ; (B): mode
1170 : 88 01                  EORA   #SOH
1172 : 26 0A                  BNE    SRRB80        ; If SOH, received header block (A=0)
1174 :                  ;
1174 : DE 52                  LDX    R1            ; Set counter
1176 : E6 04                  LDAB   SRSIZ-SRFMT,X
1178 : F7 01 C8               STAB   SRSIZ
117B : 7E 10 F3               JMP    SRRB10
117E :                  ; Completed to receive data block
117E :                  ;
117E : 4F               SRRB80 CLRA
117F : 0E               SRRB90 CLI
1180 : DE 52                  LDX    R1
1182 : 71 FB 7D               AIM    #$FF-$4,MIOSTS ; Status, stop serial communication
1185 :                  ; Recover RS232 (not change C)
1185 : 7B B0 7D               TIM    #$B0,MIOSTS   ; Broken?
1188 : 27 01                  BEQ    SRRB9A        ; Note. After CLI instruction, break...
118A : 0D                     SEC                  ; ...may be caused
118B : 7B 03 7A         SRRB9A TIM    #$3,SRSTS     ; On RS232 read running?
118E : 27 0E                  BEQ    SRRB92
1190 :                  ; Wait 250 micro sec (for serial terminal to receive character)
1190 : 36                     PSHA
1191 : 86 32                  LDAA   #50
1193 : 4A               SRRB91 DECA
1194 : 26 FD                  BNE    SRRB91
1196 : 72 04 03               OIM    #$4,PORT2     ; Select serial slave CPU
```

```
1199 : 96 5B                 LDAA   R5L            ; Recover TRCSR
119B : 97 11                 STAA   TRCSR
119D : 32                    PULA
119E : 8A 00          SRRB92 ORAA   #$0            ; For recover (Z) (unchange C)
11A0 : 39                    RTS
11A1 :                ;
11A1 :                ; Program of sending side (main device)
11A1 :                ; Get characters from keyboard and send by EPSP
11A1 :                ;
11A1 : 86 01          OPNBIS LDAA   #1             ; Driver on
11A3 : BD FF 73              JSR    SERONF
11A6 : CE 12 B8              LDX    #SCRPSD        ; Set screen packet X:data address
11A9 : C6 02                 LDAB   #SCRPSE-SCRPSD; (B):number of data
11AB : A6 00          INIT10 LDAA   0,X
11AD : A7 08                 STAA   SCRPK1-SCRPSD,X
11AF : 08                    INX
11B0 : 5A                    DECB
11B1 : 26 F8                 BNE    INIT10
11B3 :                ;
11B3 : CE 12 C0              LDX    #SCRPK1        ; Initializa screen
11B6 : BD FF 5E              JSR    SCRFNC         ; Select screen device (display controller)
11B9 :                ;
11B9 : BD FF 9A       RPEAT  JSR    KEYIN
11BC : 25 0C                 BCS    BRKRTN
11BE : B7 12 31              STAA   BUF
11C1 : 4F                    CLRA
11C2 : CE 12 2C              LDX    #SNDPKT        ; Serial transmitte
11C5 : BD FF 70              JSR    SEROUT
11C8 :                ;
11C8 : 20 EF                 BRA    RPEAT
11CA : 39             BRKRTN RTS
11CB :                ;
```

```
11CB :                 ; Program of receiving side (slave device)
11CB :                 ; Get characters from EPSP and display on the virtual screen
11CB :                 ;
11CB : 86 01           RECSID LDAA   #1             ; Driver on
11CD : BD FF 73               JSR    SERONF
11D0 : 86 01                  LDAA   #1             ; Serial master/slave mode = slave
11D2 : B7 01 CE               STAA   SRMODE
11D5 : CE 13 2E               LDX    #SCRPRD        ; Set screen packet X:data address
11D8 : C6 0E                  LDAB   #SCRPRE-SCRPRD; (B):number of data
11DA : A6 00           RECS10 LDAA   0,X
11DC : A7 92                  STAA   $92,X          ; $92 = SRCPK1 - SCRPKD
11DE : 08                     INX
11DF : 5A                     DECB
11E0 : 26 F8                  BNE    RECS10
11E2 :                 ;
11E2 : CE 12 C0               LDX    #SCRPK1        ; Initialize screen
11E5 : BD FF 5E               JSR    SCRFNC         ; Select screen device
11E8 : CE 12 C8               LDX    #SCRPK2
11EB : BD FF 5E               JSR    SCRFNC         ; Set screen size and buffer address
11EE : CE 12 CD               LDX    #SCRPK3
11F1 : BD FF 5E               JSR    SCRFNC         ; Set cursor margin
11F4 : CE 12 CF               LDX    #SCRPK4
11F7 : BD FF 5E               JSR    SCRFNC         ; Set scroll step
11FA : CE 12 D2               LDX    #SCRPK5
11FD : BD FF 5E               JSR    SCRFNC         ; Set scroll speed
1200 :                 ; Device: treat as display controller.
1200 : CC 30 20               LDD    #$3020         ; Wait to be EPSP selected
1203 : CE 00 00               LDX    #0
1206 : BD 10 3D               JSR    SRSLCT
1209 : 25 BF           RCVR10 BCS    BRKRTN
120B :                 ;
120B : CE 12 33        RCVRPT LDX    #RCVPKT        ; Receive data
```

```
120E : BD 10 DF                JSR    SERRCV
1211 : 25 B7                   BCS    BRKRTN
1213 : 27 0F                   BEQ    RCVR20        ; Error?
1215 : 81 B8                   CMPA   #$B8          ; Received EOT
1217 : 26 F2                   BNE    RCVRPT
1219 : CC 30 20                LDD    #$3020        ; Wait to be EPSP selected
121C : CE 00 00                LDX    #0
121F : BD 10 31                JSR    SRSLET
1222 : 20 E5                   BRA    RCVR10
1224 :                  ;
1224 : B6 12 38         RCVR20 LDAA   RCVPKT+5      ; Display received characters on the
1227 : BD FF 4F                JSR    DSPSCR        ; virtual screen (LCD)
122A : 20 DF                   BRA    RCVRPT
122C :                  ;
122C :                  ; Packet of send data string
122C :                  ;
122C : 00               SNDPKT FCB    $0     ; Format
122D : 30                      FCB    $30    ; SID (display controller)
122E : 20                      FCB    $20    ; DID (HC-20)
122F : 92                      FCB    $92    ; Function
1230 : 00                      FCB    0      ; Data length
1231 : 00               BUF    FCB    0      ; Data
1232 : 00                      FCB    0
1233 :                  ; Packet of receive data string
1233 : 00               RCVPKT FCB    $0     ; Format
1234 : 30                      FCB    $30    ; SID (display controller)
1235 : 20                      FCB    $20    ; DID (HC-20)
1236 : 92                      FCB    $92    ; Function
1237 : 00                      FCB    0      ; Data length
1238 :                         RMB    128    ; Data
12B8 :                  ;
12B8 :                  ; Screen packet for sending side
```

```
12B8 : 84               SCRPSD FCB    $84    ; Screen device select (display controler)
12B9 : 30                      FCB    $30
12BA :                  ;
12BA :                  ; Work area
12BA :                  SCRPSE RMB    6
12C0 : 84               SCRPK1 FCB    $84    ; Select screen device
12C1 : 22                      FCB    $22
12C2 :                         RMB    6
12C8 : 87               SCRPK2 FCB    $87    ; Set screen size and buffer address
12C9 : 13 03                   FCB    19,3
12CB : 12 D4                   FDB    SCRBUF
12CD : C3               SCRPK3 FCB    $C3    ; Set cursor margin
12CE : 04                      FCB    4
12CF :                  ;
12CF : C4               SCRPK4 FCB    $C4    ; Set scroll step
12D0 : 0A                      FCB    10     ; X
12D1 : 03                      FCB    3      ; Y
12D2 :                  ;
12D2 : CB               SCRPK5 FCB    $CB    ; Set scroll speed
12D3 : 09                      FCB    9
12D4 :                  ;
12D4 :                  SCRBUF RMB    90
132E :                  ; Screen packet for receiving side
132E : 84               SCRPRD FCB    $84    ; Screen device select (LCD)
132F : 22                      FCB    $22
1330 :                  ;
1330 : 87                      FCB    $87    ; Set screen size and buffer address
1331 : 13 03                   FCB    19,3
1333 :                  ;
1333 : 12 D4                   FDB    SCRBUF
1335 :                  ;
1335 : C3                      FCB    $C3    ; Set cursor margin
```

```
1336 : 04                             FCB    4
1337 :                         ;
1337 : C4                             FCB    $C4    ; Set scroll step
1338 : 0A                             FCB    10     ; X
1339 : 03                             FCB    3      ; Y
133A :                         ;
133A : CB                             FCB    $CB    ; Set scroll speed
133B : 09                             FCB    9
133C :                         ;
133C : =$133C                  SCRPRE EQU    *
133C :                                END
```

Chapter 5

RS-232C communication

5.1 General

The RS-232C port performs communication by the start-stop synchronization method (refer to the description of serial communication in Chapter 4). Generation of the `TXD` binary signal and read of the `RXD` binary signal are performed by software. The master MCU transmits data (`TXD`) and the slave MCU receives data (`RXD`). The slave MCU receives 1 character of data which it sends to the master MCU via the SCI. The master MCU then uses an `SCI` interrupt to store this data in the receive buffer (Figure 5.1).

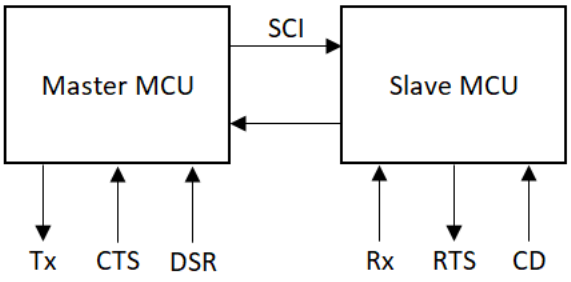


Figure 5.1: Assignment of RS-232C funtions

5.2 Data transmission method

`TXD` is controlled by port `P21` of the master MCU. When a value is set in the `OCR` and the `OCF` is set to 1, the value of the `OLVL` (bit 0 of `TCSR`) is output from `P21` (Figure 5.2).

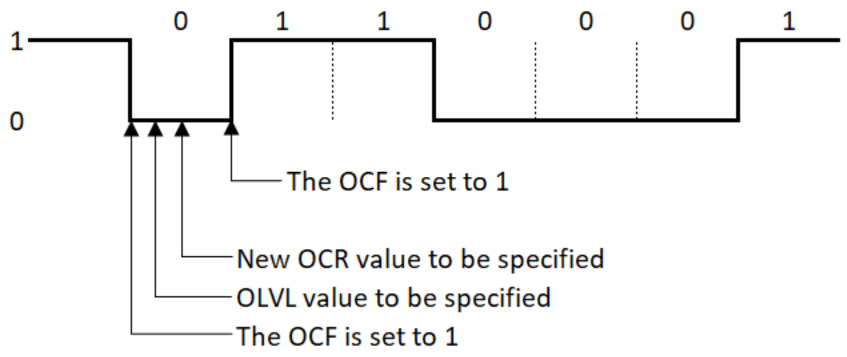


Figure 5.2: Timing of `TXD` transmission

5.3 Data reception method

Receive data is input to port `P20` of the slave MCU. Input of a start bit in `P20` is monitored.

The value of `FRC` when it takes the value specified by `IEDG` (bit 1 of `TCSR`) is set in `ICR` and this is used to measure the timing of the start bit. Based on this, the calculated center of each pulse is sampled to obtain one character of data (Figure 5.3).

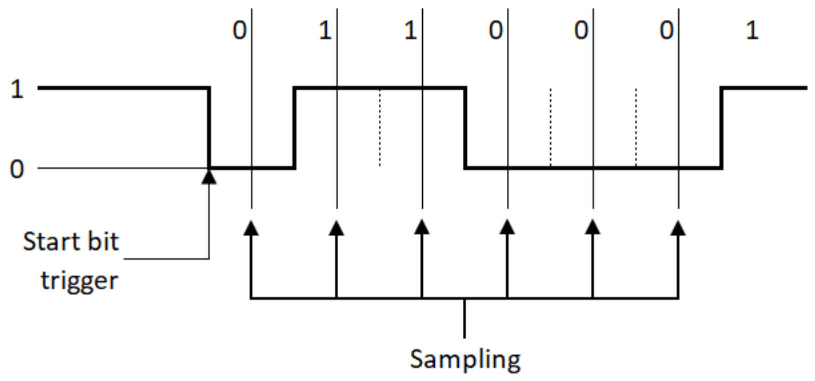


Figure 5.3: Sampling of receive data

One character of data is then transmitted to the master MCU via the SCI (Figure 5.4).

The master MCU enables receive interrupt by the SCI. The SCI receive interrupt routine stores the receive data in the receive buffer. When the buffer becomes full, an error flag is set and data received subsequent to this will be discarded. The slave MCU cancels input of data through the RS-232C

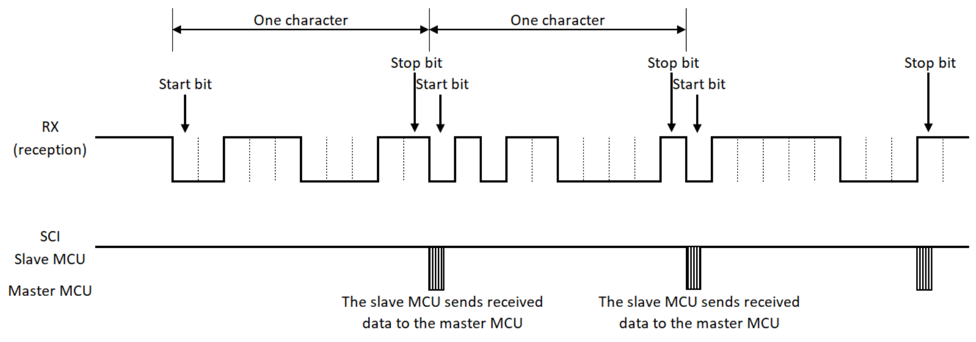


Figure 5.4: Timing of data reception

port when a command is sent to it from the master MCU.

5.4 Data communication

Data communication via the RS-232C port is performed by the following procedures.

1. Setting parameters

   Values for bit rate, word length, parity bit, stop bit length, `CD`, `RTS`, `DSR` detection, are specified by subroutine `RSMST`. This subroutine specifies the values for constants used in data communication in the I/O work area.

2. Driver ON

   Subroutine `RSONOF` turns the RS-232C driver ON. When the driver is turned ON, both `RTS` and `TXD` go low (`RTS` is turned OFF and `TXD` becomes logic `1`).

   A 10-bit preamble (logic `1`) is then output. `DTR` is directly connected to the driver power and therefore goes high (ON) when the driver is turned ON.

3. Receive buffer open

   The receive buffer in the master MCU is opened by subroutine `RSOPEN`. Once the receive buffer has been opened, the slave MCU begins sending data. The `RTS` value is set to the value specified in procedure 1 above.

4. Input of one character

Data is fetched from the receive buffer using subroutine RSGET. The data received by the slave MCU is stored in the receive buffer during SCI interrupt processing.

5. Output of one character

   Subroutine RSPUT outputs one character of data. Note that no buffer is used when outputting data.

6. Termination of data reception

   Subroutine RSCLOS terminates RS-232C data reception.

7. Driver OFF

   RSONOF is used to turn the RS-232C driver OFF.

5.5 Notes on I/O open condition

The main MCU enables SCI interrupt during RS-232C reception. When the SCI port is accessed directly, the SCI interrupt must be disabled. When the slave MCU receives new data from the SCI port, it cancels data reception from the RS-232C port. The master MCU uses subroutine SNSCOM to send a command to the slave MCU during RS-232C reception and calls subroutine CHKRS (resumption of the interrupted RS-232C data reception) upon completion of transmission of the command.

5.6 Bit rate setting

Subroutine RSMST is used to set bit rates for RS-232C transmission (110, 150, 300, 600, 1200, 2400, 4800 and 9600bps). To set a transmission speed other than one of those listed above, RSMST must be called and the desired bit rate set directly in variable RSBAUD (01AF, 01B0). This 2-byte variable indicates the number of MCU clock pulses and is set at 1000_{16} for a bit rate of 150bps. A bit rate of 75bps is therefore obtained by setting 2000_{16} in variable RSBAUD. Note that this value is used directly by the transmission subroutine so the bit rate will change as soon as the value of RSBAUD is altered.

5.7 RTS operation and carrier detection

When using a half-duplex modem, the RTS output must be changed and the carrier ON/OFF must be detected. Both RTS and the carrier ports are connected to the slave MCU. `RTS` control and `CD` detection are performed by the procedures described below.

1. `RTS` control

    - Method 1: subroutine `RSOPEN`

      `RTS` is set when reception is opened by subroutine `RSOPEN`. Reception is temporarily closed (subroutine `RSCLOS`) and the appropriate parameters are set by subroutine `RSMST` (the previously set parameters remain effective if this is not performed). Reception is then reopened by subroutine `RSOPEN`. (Figure 5.5).

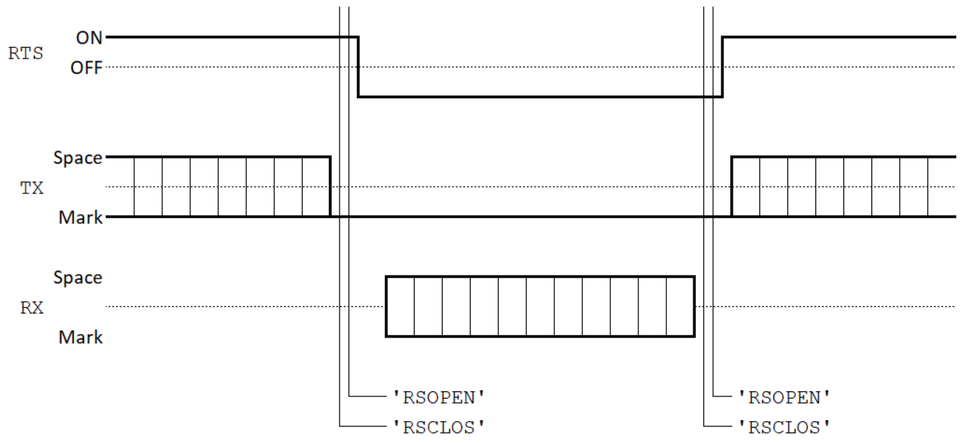


Figure 5.5: `RTS` control (1)

    - Method 2: slave MCU command

      When performing half-duplex communication, `RTS` is normally turned ON while data is being transmitted and turned OFF when data is being received. Command `4D`, sent to the slave MCU, controls the `RTS`. This command should be used to turn `RTS` ON before the start of data transmission. `RTS` should be turned OFF to open reception. (Figure 5.6).

2. Carrier detection

   When the reception is opened, the carrier status is set in port `P12` of the master MCU (port `P47` of the slave MCU actually detects the carrier status but this data is set in port `P12` of the master MCU by software). When the carrier is OFF, `P12` of the master MCU is set to `1`. When the carrier is ON, `P12` is set to `0`. Note that after reception has been

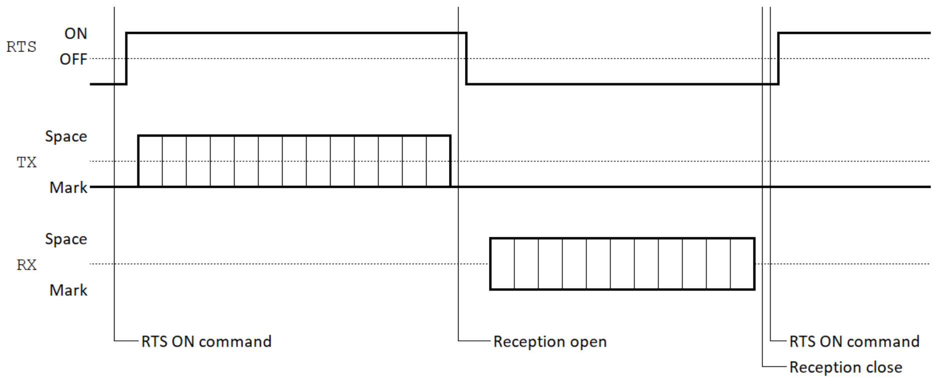


Figure 5.6: `RTS` control (2)

opened, if carrier OFF status has been detected, carrier ON will cause data reception to start but `P12` will not become `0`.

The system waits for carrier ON by the following two methods:

- Method 1: if `P12` is `1` when the reception is opened, reception is closed and then reopened. This is repeated until carrier ON is detected.
- Method 2: command `80`, which sets the value of the slave MCU port in port `P12` of the master MCU, is executed for the slave MCU until the carrier is set ON (`P12` is set to `0`). Reception is then opened.

5.8 Communications using a MODEM

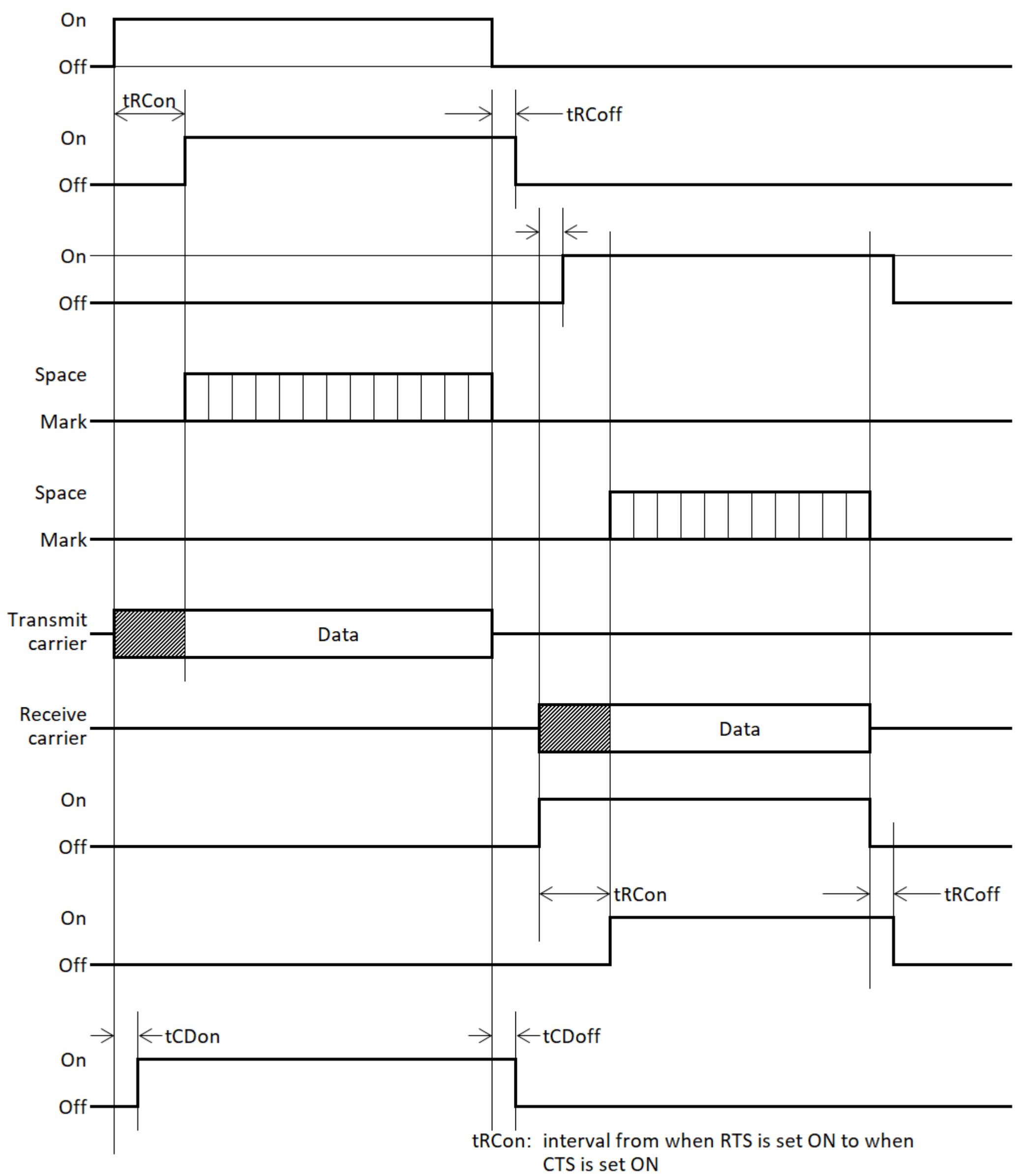
When using a MODEM, in addition to the data lines for transmission and reception, the control lines must be operated. Figure 5.7 shows the timing for a 1200bps, half-duplex MODEM.

When data communication is performed as shown in Figure 5.7, `RTS` control as well as `CTS` and `CD` detection must be confirmed.

The reception routine provides a mode in which data can be received even if no carrier has been detected. If the carrier OFF state is not of great importance, the reception can be opened in this mode and the carrier ignored.

5.8.1 1200bps reverse channels

A 1200bps MODEM may use a 75bps reverse channel. This is performed by the following two procedures.



tRCon: interval from when RTS is set ON to when CTS is set ON
tRCoff: interval from when RTS is set OFF to when CTS is set OFF
tCDon: interval from when the carrier is received to when the CD is set ON
tCDoff: interval from when the carrier is turned OFF to when the CD is set OFF

Figure 5.7: Timing of 1200bps, half-duplex MODEM

1. 1200bps transmission and 75bps reception. This is enabled by opening reception (`RSOPEN`) at 75bps and then setting the mode (`RSMOD`) at 1200bps.
2. 1200bps reception and 75bps transmission. Reception is opened at 1200bps and the bit rate is set to 75bps (`2000`$_{16}$ in variable `RSBAUD`).

Since master MCU interrupt is disabled during data transmission, data received at this time will be lost as shown in Figure 5.8.

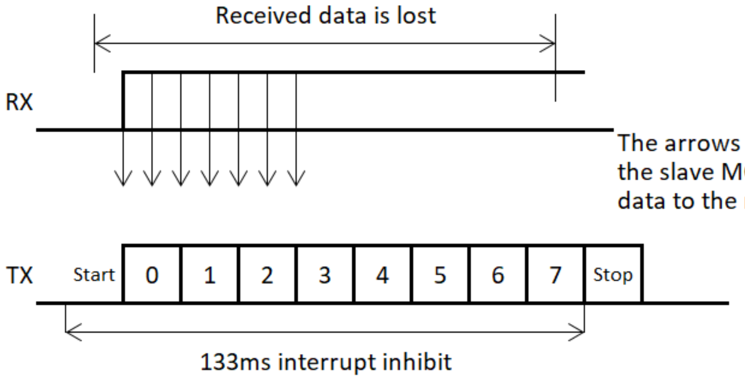

Received data is lost
RX
The arrows indicate that the slave MCU is transmitting data to the master MCU
TX
Start
0
1
2
3
4
5
6
7
Stop
133ms interrupt inhibit

Figure 5.8: Full-duplex communication at 75 and 1200bps

To protect receive data, the data transmission routine in which interrupt inhibit instruction `SEI` is omitted must be used (see Subsection 5.13.1).

5.9 Cautions for serial driver ON/OFF

1. When the driver is turned ON

   Signal rise may be unstable when the driver is turned ON as shown in Figure 5.9.

   In this case, the receiving side may receive incorrect data because it interprets the space state when the driver is turned ON as the start bit.

2. When the driver is turned OFF.

   The voltage may change as shown in Figure 5.10 when the driver is turned OFF. Again, the receiving side may interpret the resulting several tens or hundreds of bits of space states as data, resulting in erroneous data reception.

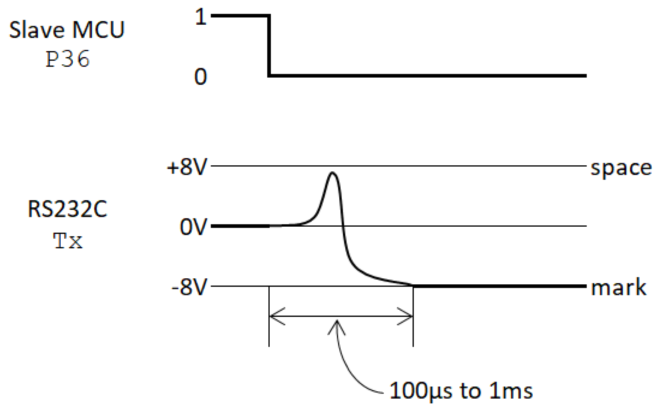


Figure 5.9: Voltage change when driver is turned ON

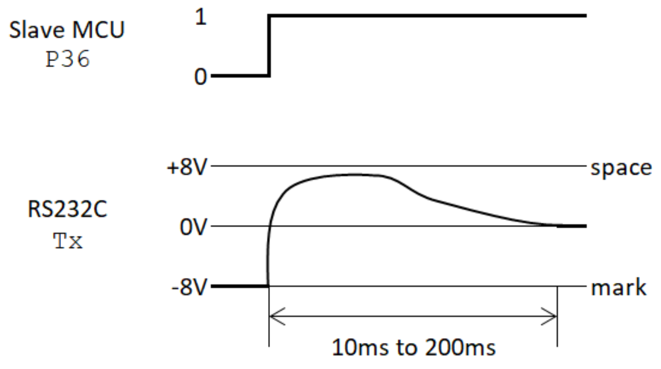


Figure 5.10: Voltage change when driver is turned OFF

The driver is turned OFF when the input through the RS-232C port is closed in BASIC. Turn the serial driver ON if you wish to leave the driver on after the RS-232C output is closed. (In terms of software, the serial and RS-232C driver are treated as separate elements. Therefore the driver will only be turned OFF when both drivers are set to OFF from software).

Press the BREAK key and check the contents of bit 7 of address `7A`. When bit 7 is `0`, the driver is ON and when it is `1`, the driver is OFF. The default value for bit 7 is `0`.

5.10 Another method of managing control lines

Since the `RTS` and `CD` control lines are connected to the slave MCU, during `RTS` control and `CD` detection there is an idle time (time required for exchanging the master MCU commands) which may cause the user inconvenience.

To avoid this, serial `POUT` and `PIN` can be used instead of `RTS` and `CTS` as control lines (Figure 5.11).

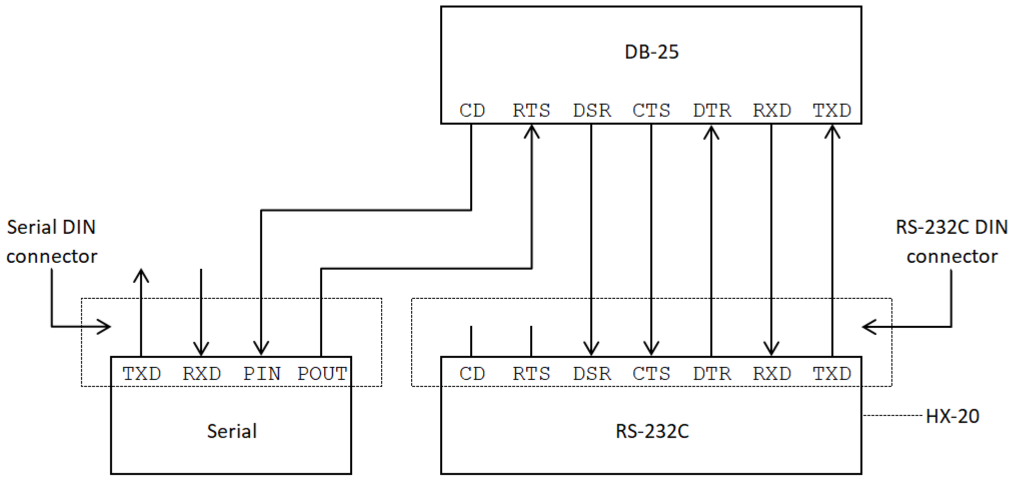


Figure 5.11: Modification of RS-232C control lines

`POUT` corresponds to bit 5 of address `26`, and is active low.

Subroutine `WRTP26` is used to set data in address `26`. `PIN` corresponds to bit 6 of port 1 and is also active low.

Note: as the floppy disk does not use `PIN` and `POUT` for serial communication, the RS-232C port can use them as control lines.

5.11 RS-232C subroutines

Subroutine name	Entry point	Description
`RSMST`	`FF8A`	Specifies the RS-232C mode. Sets values in variables `RSBITL`, `RSMODS` and `RSBAUD`. Communications with the slave MCU are not performed.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters – At entry ∗ `(A)`: mode · Bit 0 and 1: Stop bit length (`1`, `2` or `1`). · Bit 2: specifies whether or not carrier detection will be performed `0`: carrier detection. `1`: no carrier detection · Bit 3: `RTS` (`0`: OFF; `1`: ON). · Bit 4: `DRS` `0`: checks `DRS`. `1`: does not check `DRS`. · Bit 5: `CTS` `0`: checks `CTS`. `1`: does not check `CTS`. · Bits 6 and 7: parity. `0`: even. `1`: odd. `2` or `3`: none. ∗ `(B)`: bit rate and word length. · Bits 0 through 3: word length (`5`, `6`, `7` and `8`). · Bits 4 through 7: bit rate. `0`: 110bps. `1`: 150bps. `2`: 300bps. `3`: 600bps. `4`: 1200bps. `5`: 2400bps. `6`: 4800bps. `7`: 9600bps (transmission only). – At return: none. • Registers retained `(A)`, `(B)` and `(X)`. • Subroutines referenced: none. • Variables used: none.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
RSONOF	FF85	Turns ON/OFF the RS-232C driver. When bits 3 and 4 of SRSTS are off this subroutine turns the driver ON and transmits a 10-bit preamble (data logic 1). If the driver is already ON, the ON procedure will be ignored but no error will occur.
		• Parameters – At entry ∗ (A): · 0: turns OFF the driver power. · 1: turns ON the driver power. – At return ∗ (A): error code. ∗ (C): abnormal I/O flag. ∗ (Z): according to the value of (A). • Registers retained: (B) and (X). • Subroutines referenced: – SNSCOM. • Variables used: none.
RSOPEN	FF82	Opens the RS-232C input, initiates fetching data into a buffer, and exchanges commands between the master and slave MCUs. Receive data is stored in the receive buffer via the SCI (interrupt processing). When the RS-232C input is opened, RTS is set at the value specified in subroutine RSMST.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description

		• Parameters
		– At entry
		∗ `(A,B)`: receive buffer size.
		∗ `(X)`: starting address of the receive buffer.
		– At return
		∗ `(C)`: abnormal I/O flag.
		∗ `(A)`: return codes
		· `00`: RS-232C input has been correctly opened.
		· `01`: the driver is OFF.
		• Registers retained: none.
		• Subroutines referenced
		– `SNSCOM`
		– `SNSCOW`
		– `SNSDAT`
		• Variables used: none
		• Example
		In this example, a 260-byte monitor buffer is opened as the receive buffer.
		`LDAA #$0D ; Even parity, CTS/DSR check, RTS high, CD check, 1 stop bit`
		`LDAB #$27 ; 300bps, 7-bit word length`
		`JSR RSMST`
		`LDAA #1 ; Driver ON`
		`JSR RSONOF`
		`LDD #260 ; Buffer size = 260 bytes`
		`LDX #CASBUF`
		`JSR RSOPEN`

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
`RSCLOS`	`FF7F`	Closes input to the RS-232C port and sends a command to the slave MCU to terminate reception. This subroutine does not turn the driver OFF.
		• Parameters – At entry: none. – At return ∗ `(C)`: abnormal I/O flag. ∗ `(A)`: return codes `00`: RS-232C has been correctly closed (only this code is currently available). ∗ `(Z)`: according to the value of `(A)`. • Registers retained: `(B)` and `(X)`. • Variables used: none. • Subroutines referenced: none.
`RSGSTS`	`FF7C`	Inputs the value of the status register. When a receive error occurs, this subroutine fetches the error status from the slave MCU and inputs this value to the master MCU. Then, the error status of the slave MCU is cleared. Logic `1` in any bit indicates an error.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters – At entry: none – At return ∗ `(C)`: abnormal I/O flag. ∗ `(A)`: status managed by master MCU (RS-232C transmitting side) · Bit 7: `1`: receive buffer overflow. ∗ `(B)`: status manages by slave MCU (RS-232C receiving side) · Bit 0: carrier disconnection (OFF). · Bit 1: parity error. · Bit 2: overrun error. · Bit 5: receive error. • Registers retained: `(X)`. • Subroutines referenced – `SNSCOM`. – `CHKRS`. • Variables used: none.
`RSGET`	`FF79`	Fetches one character from the receive buffer. The data in the receive buffer is stored in word length + parity bit format. Once a character is fetched, the parity bit is set to `0`. This parity bit is not stored in the receive buffer if the format is 8 bits + 1 parity bit.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters – At entry: none. – At return ∗ `(C)`: abnormal I/O flag. ∗ `(A)`: received character. ∗ `(B)`: return codes · `00`: normal. · `01`: receive buffer full. · `C0`: parity error. · `C1`: carrier disconnection (OFF). **Note:** carrier disconnection (OFF) error occurs not when the carrier falls but when the buffer becomes empty. ∗ `(Z)`: according to the value of `(B)`. • Registers retained: `(X)`. • Subroutines referenced: none. • Variables used: `ROH`.
`RSPUT`	`FF76`	Transmits one character through the RS-232C port. Note that no transmit buffer is provided.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters – At entry ∗ `(A)`: output characters. If the number of bits to be transmitted is less than 8 bits, data is right-justified. The remaining bits (including the parity bit) can be any value. – At return ∗ `(C)`: abnormal I/O flag. ∗ `(B)`: return codes · `00`: normal. · `01`: no data transmitted when `DSR` is OFF. · `02`: no data transmitted when `CTS` is OFF. · `03`: no data transmitted when both `DSR` and `CTS` are OFF. ∗ `(Z)`: according to the value of `(B)`. • Registers retained: `(A)` and `(X)`. • Subroutines referenced: none. • Variables used: `R0`, `R1` and `R2H`.
`CHKRS`	`FF16`	Sends a command to the slave MCU to resume the interrupted RS-232C input.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters – At entry: none. – At reurn: none. • Registers retained: `(A)`, `(B)`, `(X)` and condition code `(CC)`. • Subroutines referenced – `RSRSRT`. • Variables used: none.

5.12 RS-232C work areas

Address (from)	(to)	Variable name	Bytes	Description
`1AF`	`1B0`	`RSBAUD`	2	RS-232C bit rates (clock cycles) • 150bps: $\texttt{1000}_{16}$ • 300bps: $\texttt{800}_{16}$
`1B1`	`1B2`	`RSCRC`	2	Polynomial expressions generated for CRC. Polynomial expression CRC-CCITT $(1 + x^5 + x^{12} + x^{16})$ equals $\texttt{8408}_{16}$ (default value). CRC-16 $(1 + x^2 + x^{15} + x^{16})$ equals $\texttt{A001}_{16}$. x^{16} is always 1, x^{15} is bit 0 and x^0 is bit 15.
`1B3`	`1B4`	`RSBCC`	2	`BCC` register for CRC check.

Continues in next page...

...continued from previous page.

Address (from)	(to)	Variable name	Bytes	Description
1B5	1B5	RSBITL	1	RS-232C word lenth (stop bit excluded). Word length must be 5, 6, 7 or 8.
1B6	1B6	RSMODS	1	RS-232C mode. • Bits 0 and 1: stop bit length (bit 1, bit 0): $(0,1) = 1; (1,0) = 2$. • Bit 2: carrier (CD) detection. – 0: carrier detection. – 1: no carrier detection. • Bit 3: RTS. – 0: RTS OFF (low level). – 1: RTS (high level). • Bit 4: DSR check. – 0: checks if DSR is OFF. – 1: does not check if DSR is OFF. • Bit 5: CTS check. – 0: checks if CTS is OFF. – 1: does not check if CTS is OFF. • Bits 6 and 7: Parity. (bit 7, bit 6) = ... – $(0,0)$: even parity. – $(0,1)$: odd parity. – $(1,x)$: no parity.

Continues in next page...

...continued from previous page.

Address (from)	(to)	Variable name	Bytes	Description
1B7	1B7	RSSTSR	1	RS-232C error status register. For all bits of this variable, logic 0 indicates normal operation and logic 1 indicates error. • Bit 0: carrier disconnection (OFF). • Bit 1: parity. • Bit 2: overrun. • Bit 3: undefined. • Bit 4: undefined. • Bit 5: receive error. • Bit 6: transmit error. • Bit 7: receive buffer overflow.
1B8	1B9	RSBFAD	2	Starting address of RS-232C receive buffer.
1BA	1BB	RSBFBT	2	Last address of Rs-232C receive buffer plus 1.
1BC	1BD	RSBFSZ	2	Size of RS-232C receive buffer (in bytes).
1BE	1BF	RSINP	2	Pointer indicating the last data stored in the RS-232C receive buffer (indicates the next address the buffer in which received data will be stored).
1C0	1C1	RSOUP	2	Pointer indicating the last data fetched from the RS-232C received buffer (indicates the next address to be fetched when data is fetched from the receive buffer).

Continues in next page...

…continued from previous page.				
Address (from)	(to)	Variable name	Bytes	Description
1C2	1C3	RSDCNT	2	Number of data in the RS-232C receive buffer (in bytes).

5.13 Sample listings

5.13.1 RS-232 send/receive data routine

```
 0 :                   ; RS232C
 0 :                   ; RS232C send/receive data routine
 0 :                   ; 2 subroutines:
 0 :                   ; 1. Get received character from RS232C received data buffer (RSGET).
 0 :                   ; 2. Transmit one character to TXD line (RSPUT).
 0 :                   ;
 0 :                   ; By K Akhane
 0 :                          PAGE   0
 0 :                          CPU    6301
 0 :                   ; MCU 6301 I/O ports
 0 : =$2               PORT1  EQU    $02    ; I/O port 1
 0 : =$3               PORT2  EQU    $03    ; I/O port 2
 0 : =$6               PORT3  EQU    $06    ; I/O port 3
 0 :                   ;
 0 :                   ; Other registers
 0 : =$9               FRC    EQU    $09    ; Free running counter
 0 : =$B               OCR    EQU    $0B    ; Output compare register
 0 : =$8               TCSR   EQU    $08    ; Time control and status register
 0 :                   ;
 0 :                   ; General registers used by I/O routine
50 :                          ORG    $50
50 : =$50              R0     EQU    R0H    ; 2 bytes register (R0H,R0L)
50 :                   R0H    RMB    1
51 :                   R0L    RMB    1
52 : =$52              R1     EQU    R1H    ; 2 bytes register (R1H,R0L)
52 :                   R1H    RMB    1
53 :                   R1L    RMB    1
54 : =$54              R2     EQU    R2H    ; 2 bytes register (R2H,R2L)
54 :                   R2H    RMB    1
55 :                   R2L    RMB    1
```

```
56 : =$56             R3     EQU    R3H    ; 2 bytes register (R3H,R3L)
56 :                  R3H    RMB    1
57 :                  R3L    RMB    1
7A :                         ORG    $7A
7A :                  SRSTS  RMB    1      ; Serial status
7B :                                       ; Bit 0,1: RS232 mode (00: stop; 01: interrupt read
7B :                                       ;                      02: read one character)
7B :                                       ; Bit 2: execute/pause (0: on execute; 1: pause)
7B :                                       ; Bit 3: RS232 driver (0: off; 1: driver on)
7B :                                       ; Bit 4: serial driver (0: off; 1: driver on)
7B :                                       ; Bit 5,6,7: CPU serial receive interrupt mode
7B :                                       ;         0: external cassette read
7B :                                       ;         1: micro cassette read
7B :                                       ;         2: RS232C read
7B :                                       ;         3: read from serial communication
7B :                                       ;         4: external cassette write
7B :                                       ;         5: micro cassette write
7B :                                       ;         6, 7: undefined for write
7B :                  RUNMOD RMB    1      ; Run mode ($80: BASIC; $00: system)
7C :                  SIOSTS RMB    1      ; Slave I/O status (each bit 0:off; 1:on)
7D :                                       ; Bit 0: printer
7D :                                       ; Bit 1: external cassette
7D :                                       ; Bit 2: internal micro cassette
7D :                                       ; Bit 3: RS232C on (read)
7D :                                       ; Bit 4: speaker on
7D :                                       ; Bit 5: ROM cassette
7D :                                       ; Bit 6: bar code reader
7D :                                       ; Bit 7: break slave CPU (0: on execute
7D :                                       ;                         1: broken by interrupt)
7D :                  MIOSTS RMB    1      ; Main I/O status (each bit 0:off; 1:on)
7E :                                       ; Bit 0: LCD on read/write characters
7E :                                       ; Bit 1: now sending command to slave CPU
```

```
63/      7E :                                   ; Bit 2: now sending data to serial line
64/      7E :                                   ; Bit 3: on clock interrupt
65/      7E :                                   ; Bit 4: (power fail)
66/      7E :                                   ; Bit 5: (off power switch)
67/      7E :                                   ; Bit 6: on pause key
68/      7E :                                   ; Bit 7: on break key
69/      7E :                     ; Work area
70/     1AF :                            ORG    $1AF
71/     1AF :                     ; RS232C work area
72/     1AF : =$1AF               RSWKTP EQU    RSBAUD ; RS232C work top address
73/     1AF :                     RSBAUD RMB    2      ; RS232C bit rate (number of clock cycles)
74/     1B1 :                     RSCRC  RMB    2      ; RS232C generating polynomial
75/     1B3 :                     RSBCC  RMB    2      ; RS232C BCC register
76/     1B5 :                     RSBITL RMB    1      ; RS232C bit length (5, 6, 7 or 8)
77/     1B6 :                     RSMODS RMB    1      ; RS232C mode
78/     1B7 :                                   ; Bit 0,1: number of stop bits
79/     1B7 :                                   ; Bit 2: carrier detect mask (0: check; 1: mask)
80/     1B7 :                                   ; Bit 3: clear to send (0: low; 1: high)
81/     1B7 :                                   ; Bit 4: DSR (0: check; 1: no check)
82/     1B7 :                                   ; Bit 5: CTS (0: check; 1: no check)
83/     1B7 :                                   ; Bit 6,7: parity (00: even; 01: odd;
84/     1B7 :                                   ;                  10, 11: no parity)
85/     1B7 :                     ; RS232C buffer pointer
86/     1B7 :                     RSSTSR RMB    1      ; RS232C status register
87/     1B8 :                                   ; Bit 0: carrier detect  (0: normal; 1: error)
88/     1B8 :                                   ; Bit 1: parity          (0: normal; 1: error)
89/     1B8 :                                   ; Bit 2: overrun         (0: normal; 1: error)
90/     1B8 :                                   ; Bit 5: read error      (0: normal; 1: error)
91/     1B8 :                                   ; Bit 6: write error     (0: normal; 1: error)
92/     1B8 :                                   ; Bit 7: buffer overflow (0: normal; 1: overflow)
93/     1B8 :                     ;
94/     1B8 :                     RSBFAD RMB    2      ; RS232C read buffer address
```

```
1BA :                 RSBFBT RMB    2      ; RS232C read buffer bottom address + 1
1BC :                 RSBFSZ RMB    2      ; RS232C read buffer size (0001 - FFFF)
1BE :                 RSINP  RMB    2      ; Pointer to where next received character will be stored
1C0 :                 RSOUP  RMB    2      ; Pointer to where next character will be loaded
1C2 :                 RSDCNT RMB    2      ; Number of data in the buffer
1C4 :                 ;
1C4 :                 ; RS232C: get one character from receive buffer
1C4 :                 ;      1: get one character from RS232C receive buffer
1C4 :                 ;      2: if bit length < 8 and parity check mode, do parity check and set
1C4 :                 ;         return core
1C4 :                 ; Parameter
1C4 :                 ; On entry: none
1C4 :                 ; On exit
1C4 :                 ;   (A): character (without parity bit)
1C4 :                 ;   (B): status - $01: receive buffer is empty
1C4 :                 ;                 $00: normal (MSB 1: error; 0: normal)
1C4 :                 ;                 $C0: parity error
1C4 :                 ;                 $C1: CD error (carrier down)
1C4 :                 ;   (C): slave status (0: normal; 1: error)
1C4 :                 ;   Sets Z, N flags depending on value of (B) register
1C4 :                 ; Register preserve X
1C4 :                 ; Work use as register:
1C4 :                 ;   R0H: effective bits as data (bit length=7, then $7F;
1C4 :                 ;                               bit length=8, then $FF)
1C4 :                 ;
1C4 : C6 01           RSGET  LDAB   #$01          ; Preset "buffer empty" code
1C6 : 0D                     SEC                  ; Preset error I/O flags
1C7 : 7B B0 7D               TIM    #$B0,MIOSTS   ; Error I/O?
1CA : 26 42                  BNE    RSIN23
1CC : 3C                     PSHX
1CD :                 ;
1CD : FE 01 C2               LDX    RSDCNT        ; Are there data in the buffer?
```

```
1D0 : 27 3D                BEQ    RSIN25        ; (B): 1
1D2 :               ; Set effective bits to ROH
1D2 : FC 01 B5             LDD    RSBITL        ; (B): RSMODS
1D5 : 7F 00 50             CLR    ROH           ; (A): bit length
1D8 : 0D            RSIN1A SEC
1D9 : 79 00 50             ROL    ROH           ; ROH <- $7F if B=7; $FF if B=8
1DC : 4A                   DECA
1DD : 26 F9                BNE    RSIN1A
1DF :               ;
1DF : 0F                   SEI                  ; If RS232 received interrupt is caused, the
1E0 : FE 01 C0             LDX    RSOUP         ; pointer may be destroyed
1E3 : A6 00                LDAA   0,X           ; (A): data
1E5 : 08                   INX
1E6 : BC 01 BA             CPX    RSBFBT        ; If pointer shows bottom address + 1 of the
1E9 : 26 03                BNE    RSIN10        ; buffer, pointer must be set to top address.
1EB : FE 01 B8             LDX    RSBFAD
1EE : FF 01 C0      RSIN10 STX    RSOUP
1F1 : FE 01 C2             LDX    RSDCNT
1F4 : 09                   DEX
1F5 : FF 01 C2             STX    RSDCNT        ; Data counter <- current value - 1
1F8 : 0E                   CLI
1F9 :               ; Parity error check
1F9 : 58                   ASLB                 ; Parity check mode?
1FA : 25 0F                BCS    RSIN15        ; Mode = check parity?
1FC : 58                   ASLB
1FD : 16                   TAB                  ; (B) <- data; (C) <- parity mode (0: even)
1FE : 94 50                ANDA   ROH           ; Take data bits (ignore parity bit)
200 : 24 02         RSIN11 BCC    RSIN12
202 : C8 80                EORB   #$80
204 : 58            RSIN12 ASLB
205 : 26 F9                BNE    RSIN11
207 : 56                   RORB                 ; Bit7, bit6 <- (C)
```

```
208 : 57                        ASRB
209 : 20 01                     BRA    RSIN20        ; Parity error = $C0
20B :                    ;
20B : 5F                 RSIN15 CLRB                 ; Normal return
20C :                    ; Buffer is empty
20C : 5D                 RSIN20 TSTB                 ; Clear (C), set (Z)
20D : 38                        PULX
20E : 39                 RSIN23 RTS
20F :                    ; Buffer is empty; is carrier down?
20F : 7B 04 7A           RSIN25 TIM    #$4,SRSTS     ; On pause?
212 : 26 F8                     BNE    RSIN20
214 : 7B 04 02                  TIM    #$4,PORT1     ; SFLAG = on?
217 : 27 F3                     BEQ    RSIN20
219 : C6 C1                     LDAB   #$C1          ; CD error
21B : 20 EF                     BRA    RSIN20
21D :                    ;
21D :                    ;
21D :                    ; Send one transmitted character subroutines
21D :                    ; Parameter
21D :                    ; On entry
21D :                    ;   Transmitted character
21D :                    ; On exit
21D :                    ;   (B): bit 0 (1:DSR low) character is not sent
21D :                    ;        bit 1 (1:CTS low) character is not sent
21D :                    ;        bit 2 - 7 (always 0)
21D :                    ;   (Z): depends on value of (B)
21D :                    ;   (C): 0: normal; 1: I/O error
21D :                    ; Register preserve A, X
21D :                    ; Work use as register
21D :                    ;   R0H: parity bit (LSB)
21D :                    ;   R0L: "with parity bit" flag (0: yes; 1: no)
21D :                    ;   R1H: save data
```

```
21D :                   ;   R1L: bit length
21D :                   ;
21D :                   ; Note: OCR is used, and OCR is used by key routine either.
21D :                   ;
21D : 0D                RSPUT  SEC                  ; Preset I/O error flag
21E : 7B B0 7D                 TIM    #$B0,MIOSTS   ; I/O error?
221 : 26 0F                    BNE    SNDR04
223 :                   ; Check DSR, CTS
223 : F6 01 B6                 LDAB   RSMODS        ; Take mode (DSR CTS bits)
226 : 57                       ASRB                 ; RSMODS (DSR: bit 4; CTS: bit 5) mask = 1
227 : 57                       ASRB
228 : 57                       ASRB
229 : 57                       ASRB                 ; PORT1 (DSR: bit 0; CTS: bit 1) normal = low
22A : 53                       COMB
22B : D4 02                    ANDB   PORT1         ; Check DSR, CTS
22D : C4 03                    ANDB   #$3
22F : 27 02                    BEQ    SNDR05
231 : 0C                       CLC                  ; CTS, DST low (error)
232 : 39                SNDR04 RTS
233 :                   ;
233 : 36                SNDR05 PSHA
234 : 97 52                    STAA   R1H
236 : 3C                       PSHX
237 : CE 01 AF                 LDX    #RSWKTP       ; (X): top RAM address of work area for RS232C
23A : 0F                       SEI                  ; Disable interrupt
23B : A6 06                    LDAA   RSBITL-RSWKTP,X
23D : 97 53                    STAA   R1L
23F : 4F                       CLRA
240 : E6 07                    LDAB   RSMODS-RSWKTP,X ; RSMODS (bit 7: with parity flag;
242 : 05                       ASLD                 ;          bit 6: even or odd)
243 : 97 51                    STAA   R0L           ; R0L: number of parity bits (R0L: 0 or 1)
245 : 4F                       CLRA
```

```
246 : 05                         ASLD
247 : 97 50                      STAA   R0H            ; LSB <- parity
249 :                     ;
249 : 7B 40 08            SNDR20 TIM    #$40,TCSR      ; OCR overflow?
24C : 26 0B                      BNE    SNDR30
24E :                     ; Not overflow
24E : DC 0B                      LDD    OCR            ; "Time till next edge" < 1.6*$20 microsecs?
250 : 93 09                      SUBD   FRC            ; Yes, then wait OCR overflow, now begin "start
252 : 83 00 20                   SUBD   #$20           ;   bit"
255 : 2B F2                      BMI    SNDR20         ; No, then wait time of start bit
257 : 20 07                      BRA    SNDR40
259 :                     ; OCR over, set next time
259 : DC 09               SNDR30 LDD    FRC            ; Set time of start bit
25B : C3 00 20                   ADDD   #$20
25E : DD 0B                      STD    OCR
260 : 71 FE 08            SNDR40 AIM    #$FF-$01,TCSR ; Set "low"
263 :                     ;
263 : 7B 40 08            SNDR45 TIM    #$40,TCSR      ; Wait until overflow
266 : 27 FB                      BEQ    SNDR45
268 :                     ; Set next data bit
268 : 5F                         CLRB
269 : 77 00 52                   ASR    R1H
26C : 59                         ROLB                  ; (B) 0 or 1
26D : 26 05                      BNE    SNDR50
26F :                     ; Set 0
26F : 71 FE 08                   AIM    #$FF-$1,TCSR
272 : 20 06                      BRA    SNDR53         ; Parity is not changed
274 :                     ; Set 1
274 : 72 01 08            SNDR50 OIM    #$1,TCSR
277 : 75 01 50                   EIM    #$1,R0H        ; Compute parity
27A :                     ;
27A : E8 05               SNDR53 EORB   RSBCC+1-RSWKTP,X ; Compute CRC
```

```
27C : A6 04                 LDAA   RSBCC-RSWKTP,X
27E : 04                    LSRD
27F : 24 04                 BCC    SNDR54
281 : A8 02                 EORA   RSCRC-RSWKTP,X
283 : E8 03                 EORB   RSCRC+1-RSWKTP,X
285 : ED 04          SNDR54 STD    RSBCC-RSWKTP,X
287 :                ; Set next time
287 : DC 0B                 LDD    OCR
289 : E3 00                 ADDD   RSBAUD-RSWKTP,X
28B : DD 0B                 STD    OCR
28D : 7A 00 53              DEC    R1L           ; Finished?
290 : 26 D1                 BNE    SNDR45
292 :                ; Add parity?
292 : 96 51                 LDAA   R0L
294 : 26 0A                 BNE    SNDR60
296 : D6 50                 LDAB   R0H           ; Set parity (R1H <- R0H)
298 : 4C                    INCA                 ; "Add parity" flag <- "none" (R0L <- 0)
299 : DD 51                 STD    R0L
29B : 7C 00 53              INC    R1L           ; Bit count <- 1
29E : 20 C3                 BRA    SNDR45
2A0 :                ;
2A0 :                ; Add stop bits
2A0 : 7B 40 08       SNDR60 TIM    #$40,TCSR     ; Wait until start of last bit
2A3 : 27 FB                 BEQ    SNDR60
2A5 :                ;
2A5 : DC 0B                 LDD    OCR
2A7 : E3 00                 ADDD   RSBAUD-RSWKTP,X
2A9 : DD 0B                 STD    OCR
2AB :                ;
2AB : 72 01 08              OIM    #$1,TCSR      ; Stop bit
2AE : 7B 40 08       SNDR70 TIM    #$40,TCSR     ; Wait until start time of stop bit
2B1 : 27 FB                 BEQ    SNDR70
```

```
287/     2B3 :                  ;
288/     2B3 : EE 06                   LDX    RSMODS-RSWKTP-1,X
289/     2B5 : 18                      XGDX                  ; (X): OCR last time
290/     2B6 : C4 03                   ANDB   #$3            ; (B): MSMODS (LS 3 bits: number of stop bits)
291/     2B8 : 26 01                   BNE    SNDR80
292/     2BA : 5C                      INCB                  ; If 0, 1 stop bit
293/     2BB : 4F               SNDR80 CLRA                  ; (X): number of stop bits
294/     2BC : 18                      XGDX
295/     2BD : F3 01 AF         SNDR90 ADDD   RSBAUD         ; (A,B): high bit time
296/     2C0 : 09                      DEX
297/     2C1 : 26 FA                   BNE    SNDR90
298/     2C3 :                  ;
299/     2C3 : DD 0B                   STD    OCR
300/     2C5 :                  ;
301/     2C5 : 38                      PULX
302/     2C6 : 32                      PULA
303/     2C7 : 0E                      CLI                   ; If received key interrupt, key sampling time is
304/     2C8 :                  ;                            ; not punctual
305/     2C8 : 5F                      CLRB
306/     2C9 : 39                      RTS
307/     2CA :                  ;
308/     2CA :                         END
```

5.13.2 Terminal mode without hard copy

```
  1/       0 :                  ; TERM
  2/       0 :                  ; TSS terminal mode
  3/       0 :                  ; 300bps, full duplex, without hard copy
  4/       0 :                  ;
  5/       0 :                         PAGE   0
  6/       0 :                         CPU    6301
  7/    1000 :                         ORG    $1000
```

```
1000 :                   ;
1000 :                   ; Example of terminal mode
1000 :                   ;
1000 : =$FF4F            DSPSCR EQU    $FF4F
1000 : =$FF5E            SCRFNC EQU    $FF5E
1000 : =$FF85            RSONOF EQU    $FF85
1000 : =$FF88            RSMST  EQU    $FF88
1000 : =$FF82            RSOPEN EQU    $FF82
1000 : =$FF7F            RSCLOS EQU    $FF7F
1000 : =$FF79            RSGET  EQU    $FF79
1000 : =$FF76            RSPUT  EQU    $FF76
1000 : =$FF9A            KEYIN  EQU    $FF9A
1000 : =$FF9D            KEYSTS EQU    $FF9D
1000 :                   ;
1000 :                   ; Initialize
1000 : CC 84 22                 LDD    #$8422       ; Construct screen packet
1003 : FD 10 5B                 STD    SCRPK1
1006 : 86 87                    LDAA   #$87
1008 : B7 10 5D                 STAA   SCRPK2
100B : CC 13 03                 LDD    #$1303
100E : FD 10 5E                 STD    SCRPK2+1
1011 : CC 14 00                 LDD    #$1400
1014 : FD 10 60                 STD    SCRPK2+3
1017 : CE 10 5B                 LDX    #SCRPK1      ; Initialize screen
101A : BD FF 5E                 JSR    SCRFNC
101D : CE 10 5D                 LDX    #SCRPK2
1020 : BD FF 5E                 JSR    SCRFNC
1023 : CC 3D 27                 LDD    #$3D27       ; Set mode (stop:1 CD:no-check RTS:on Parity:E
1026 :                   ;                          ;           7 bits length, 300bps)
1026 : BD FF 88                 JSR    RSMST
1029 : 86 01                    LDAA   #1           ; RS232C driver on
102B : BD FF 85                 JSR    RSONOF
```

```
102E : FE FF DC            LDX    $FFDC         ; (X): buffer address (system buffer)
1031 : CC 01 04            LDD    #260          ; (A,B): buffer size
1034 : BD FF 82            JSR    RSOPEN        ; Receive open
1037 :              ;
1037 : BD FF 9D     REDKEY JSR    KEYSTS        ; Accept from keyboard?
103A : 25 1E               BCS    BRKRTN        ; If BREAK key is pressed, return (in BASIC
103C : 27 09               BEQ    RCVRS         ;   mode)
103E :              ; Accepted character from KB
103E : BD FF 9A            JSR    KEYIN
1041 : BD FF 76            JSR    RSPUT         ; Transmit accepted character
1044 : BD FF 4F            JSR    DSPSCR        ; Display accepted character to virtual screen
1047 : FE FF D8     RCVRS  LDX    $FFD8         ; Are there received characters in the buffer?
104A : EC 00               LDD    0,X
104C : 27 E9               BEQ    REDKEY
104E : BD FF 79            JSR    RSGET
1051 : 81 7F               CMPA   #$7F
1053 : 24 E2               BCC    REDKEY        ; Ignore 7F - FF characters
1055 : BD FF 4F            JSR    DSPSCR        ; Display received character to virtuual screen
1058 : 20 DD               BRA    REDKEY
105A :              ;
105A : 39           BRKRTN RTS
105B :              ; Virtual screen packet
105B : 84           SCRPK1 FCB    $84           ; Select screen device (LCD)
105C : 22                  FCB    $22
105D : 87           SCRPK2 FCB    $87           ; Set screen size and buffer address
105E : 13 03               FCB    19,3
1060 : 14 00               FDB    $1400
1062 :              ;
1062 :                     END
```

5.13.3 Terminal mode with hard copy

```
0 :                     ; TERM2
0 :                     ; TSS terminal mode
0 :                     ; Example of terminal mode
0 :                     ;   300bps full duplex terminal mode (1200bps)
0 :                     ;   Virtual screen size = 20x4
0 :                     ;   Received and transmitted characters are able to print to serial
0 :                     ;   printer (MP-80,...). The connector for hard copy is "serial".
0 :                     ;   Hard copy routine is included in interrupt procedure.
0 :                     ;
0 :                     ; Cable
0 :                     ;   1. For connect to modem (CP-20)
0 :                     ;      Optimal cable
0 :                     ;   2. For hard copy
0 :                     ;      HC-20 serial (DIN 5 pins)   MP-80 serial (DB-25)
0 :                     ;        1 (ground) -----------------  7 (ground)
0 :                     ;        2 (PTX)    -----------------  3 (RXD)
0 :                     ;        3 (PRX)    -----------------  2 (TXD)
0 :                     ;        4 (POUT)   -----------------  6 (DSR)
0 :                     ;        5 (PIN)    ----------------- 20 (DTR)
0 :                     ;        FG         -----------------  1 (protective ground)
0 :                     ;
0 :                     ;
0 :                     ; Operation
0 :                     ;   PF1 key: start hard copy
0 :                     ;   PF2 key: stop hard copy
0 :                     ;   PF3 key: 1200bps (display monitor (received character) = off)
0 :                     ;   PF4 key: 300bps
0 :                     ;   PF5 key: quit
0 :                     ;   PF6 key: monitor display on
0 :                     ;   PF7 key: monitor display off
```

```
31/       0 :                   ;   PF8 key: Esc 'I'+$20 '0'
32/       0 :                   ;
33/       0 :                   ; 1200bps full duplex terminal procedure
34/       0 :                   ;   1. PF3 (1200bps)
35/       0 :                   ;   2. PF6 (monitor display off, hard copy on)
36/       0 :                   ;   3. (PF8 ?????)
37/       0 :                   ;
38/       0 :                         PAGE   0
39/       0 :                         CPU    6301
40/       0 :                   ; Subroutine entry point
41/       0 : =$FF4F            DSPSCR EQU    $FF4F  ; Display one character to virtual screen
42/       0 : =$FF5E            SCRFNC EQU    $FF5E  ; Virtual screen function
43/       0 : =$FF85            RSONOF EQU    $FF85  ; RS232C driver on/off
44/       0 : =$FF88            RSMST  EQU    $FF88  ; Set RS232C parameters
45/       0 : =$FF73            SERONF EQU    $FF73  ; Serial driver on/off
46/       0 : =$FF82            RSOPEN EQU    $FF82  ; Open RS232C receive
47/       0 : =$FF7F            RSCLOS EQU    $FF7F  ; Close RS232C receive
48/       0 : =$FF79            RSGET  EQU    $FF79  ; Get RS232C one character
49/       0 : =$FF76            RSPUT  EQU    $FF76  ; Send RS232C one character
50/       0 : =$FF9A            KEYIN  EQU    $FF9A  ; Get one character from keyboard buffer
51/       0 : =$FF9D            KEYSTS EQU    $FF9D  ; Get number of characters in the key buffer
52/       0 : =$FF25            MENU   EQU    $FF25  ; Menu
53/       0 :                   ; Constants or registers
54/       0 : =$11              TRCSR  EQU    $11    ; Transmit/receive control register
55/       0 : =$13              STDR   EQU    $13    ; Serial transmit data register
56/       0 : =$12              SRDR   EQU    $12    ; Serial receive data register
57/       0 : =$8               TCSR   EQU    $08    ; Timer control and status register
58/       0 : =$B               OCR    EQU    $0B    ; Output compare register
59/       0 : =$9               FRC    EQU    $09    ; Free running counter
60/       0 : =$10              RMCR   EQU    $10    ; Rate and mode control register
61/       0 :                   ;                    ; 04: 38.4kbps, 05:4.4 kbps
62/       0 : =$2               PORT1  EQU    $02    ; I/O PORT1
```

```
   0 : =$3               PORT2  EQU    $03    ; I/O PORT2
   0 : =$1000            BUFSIZ EQU    4096   ; Buffer size for printer
   0 : =$C8              SCBSIZ EQU    200    ; Buffer size for screen
   0 : =$1000            RSBSIZ EQU    4096   ; Buffer size for RS232C
   0 : =$1               ECHODT EQU    1      ; Terminal mode = "echo character"?
   0 :                   ;                    ; 0: yes; 1: no
   0 : =$109             SERVCT EQU    $109   ; SCI receive interrupt address
   0 :                   ;
1000 :                          ORG    $1000
1000 :                   ;
1000 :                   ; Initialize
1000 : 86 01                    LDAA   #ECHODT
1002 : B7 11 E3                 STAA   ECHO
1005 : CE 11 C7                 LDX    #SCRPKD       ; Set screen packet X: data address
1008 : C6 0E                    LDAB   #SCRPKE-SCRPKD; (B): number of data
100A : A6 00             INIT10 LDAA   0,X
100C : A7 0E                    STAA   SCRPK1-SCRPKD,X
100E : 08                       INX
100F : 5A                       DECB
1010 : 26 F8                    BNE    INIT10
1012 :                   ;
1012 : CE 11 D5                 LDX    #SCRPK1       ; Initialize screen
1015 : BD FF 5E                 JSR    SCRFNC        ; Select screen device
1018 : CE 11 D7                 LDX    #SCRPK2
101B : BD FF 5E                 JSR    SCRFNC        ; Set screen size and buffer address
101E : CE 11 DC                 LDX    #SCRPK3
1021 : BD FF 5E                 JSR    SCRFNC        ; Set cursor margin
1024 : CE 11 DE                 LDX    #SCRPK4
1027 : BD FF 5E                 JSR    SCRFNC        ; Set scroll step
102A : CE 11 E1                 LDX    #SCRPK5
102D : BD FF 5E                 JSR    SCRFNC        ; Set scroll speed
1030 :                   ;
```

```
1030 : 86 01                     LDAA   #1            ; Monitor on
1032 : B7 11 EA                  STAA   MONFLG
1035 :                    ;
1035 : CC 11 F1                  LDD    #BUF          ; Set buffer pointer for hard copy
1038 : FD 11 EB                  STD    BPIN
103B : FD 11 ED                  STD    BPOUT
103E : CC 00 00                  LDD    #0            ; Character counter = 0
1041 : FD 11 EF                  STD    BUFCNT
1044 : B7 11 E4                  STAA   PRTFLG        ; Hard copy = "no"
1047 :                    ; Rewrite serial receive interrupt vector
1047 :                    ; Note: if we want to send a character to the printer, we may detach
1047 :                    ;       slave MPU while 20ms after we got the character from slave MCU.
1047 :                    ;
1047 : FC 01 0A                  LDD    SERVCT+1      ; Save vector address
104A : FD 11 E5                  STD    SERADR
104D : CC 11 46                  LDD    #SERINT       ; Write new interrupt address
1050 : FD 01 0A                  STD    SERVCT+1
1053 :                    ;
1053 : CC 3D 27                  LDD    #$3D27        ; Set mode (stop: 1, CD: no-check, RTS: on,
1056 :                    ;                           ;           parity: E, 7 bits length, 300bps)
1056 : FD 11 E8                  STD    RSPARM        ; Save parameters
1059 : BD FF 88                  JSR    RSMST
105C : 86 01                     LDAA   #1            ; RS232C driver on
105E : BD FF 85                  JSR    RSONOF
1061 : 86 01                     LDAA   #1            ; Serial driver on
1063 : BD FF 73                  JSR    SERONF
1066 :                    ;
1066 : CE 22 B9           INIT30 LDX    #RSBUFF       ; (X): buffer address (system buffer)
1069 : CC 10 00                  LDD    #RSBSIZ       ; (A,B): buffer size
106C : BD FF 82                  JSR    RSOPEN        ; Open to receive RS232C
106F : BD FF 9D           REDKEY JSR    KEYSTS        ; Accept from keyboard?
1072 : 25 7E                     BCS    BRKRTN        ; If Break key is pressed, return (in BASIC
1074 :                 ;                           ;   mode)
1074 : 27 27                  BEQ    RCVRS
1076 :                 ; Accepted character from keyboard
1076 : BD FF 9A               JSR    KEYIN
1079 : 81 FE                  CMPA   #$FE          ; Function codes?
107B : 26 13                  BNE    GETKEY
107D :                 ;
107D :                 ; Function keys
107D : C0 F1                  SUBB   #$F1          ; F1 - F10?
107F : 25 1C                  BCS    RCVRS         ; No, ignore
1081 : C1 0A                  CMPB   #$A
1083 : 24 18                  BCC    RCVRS
1085 : 58                     ASLB
1086 : CE 10 D6               LDX    #FNCTBL       ; Get function address
1089 : 3A                     ABX
108A : EE 00                  LDX    0,X           ; (X) <- entry point of each subroutine
108C : AD 00                  JSR    0,X
108E : 20 0D                  BRA    RCVRS
1090 : BD FF 76        GETKEY JSR    RSPUT         ; Transmit character to RS232C
1093 : F6 11 E3               LDAB   ECHO          ; Echo?
1096 : 27 02                  BEQ    GETK10
1098 : 8D 1C                  BSR    PSHCHR        ; Push received character to stack
109A : BD FF 4F        GETK10 JSR    DSPSCR        ; Display character to virtual screen
109D :                 ;
109D : CE FF D8        RCVRS  LDX    #$FFD8        ; Are there characters in the RS232C buffer?
10A0 : EC 00                  LDD    0,X
10A2 : 27 0F                  BEQ    RCVR80
10A4 : BD FF 79               JSR    RSGET
10A7 : 81 7F                  CMPA   #$7F
10A9 : 24 08                  BCC    RCVR80        ; Ignore 7F - FF characters
10AB : F6 11 EA               LDAB   MONFLG        ; Display on?
10AE : 27 03                  BEQ    RCVR80
```

```
10B0 :                 ;
10B0 : BD FF 4F        RCVR10 JSR    DSPSCR        ; Display character to virtual screen
10B3 : 7E 10 6F        RCVR80 JMP    REDKEY
10B6 :                 ;
10B6 :                 ; Push received character to print stack
10B6 :                 ; On entry
10B6 :                 ;   (A): character
10B6 :                 ; On exit
10B6 :                 ; Register preserve A, B
10B6 :                 ;
10B6 : 7D 11 E4        PSHCHR TST    PRTFLG        ; Hard copy = yes?
10B9 : 27 1A                  BEQ    PSHC80
10BB : 0F                     SEI
10BC : FE 11 EB               LDX    BPIN
10BF : A7 00                  STAA   0,X
10C1 : 08                     INX
10C2 : 8C 01 F1               CPX    #BUF-BUFSIZ
10C5 : 26 03                  BNE    PSHC10
10C7 : CE 11 F1               LDX    #BUF
10CA : FF 11 EB        PSHC10 STX    BPIN
10CD : FE 11 EF               LDX    BUFCNT
10D0 : 08                     INX
10D1 : FF 11 EF               STX    BUFCNT
10D4 : 0E                     CLI
10D5 : 39              PSHC80 RTS
10D6 :                 ;
10D6 :                 ; Function key procedure table
10D6 :                 ;
10D6 : 10 EA           FNCTBL FDB    PFKY10 ; PF1  (hard copy on)
10D8 : 10 EE                  FDB    PFKY20 ; PF2  (hard copy off)
10DA : 10 F3                  FDB    PFKY30 ; PF3  (1200bps)
10DC : 11 08                  FDB    PFKY40 ; PF4  (300bps)
```

```
10DE : 11 30                  FDB    PFKY50 ; PF5  (quit)
10E0 : 11 15                  FDB    PFKY60 ; PF6  (monitor on)
10E2 : 11 19                  FDB    PFKY70 ; PF7  (monitor off)
10E4 : 11 23                  FDB    PFKY80 ; PF8  (Esc 'I'+$20 '1')
10E6 : 11 45                  FDB    INVLKY ; PF9  (undefined)
10E8 : 11 45                  FDB    INVLKY ; PF10 (undefined)
10EA :                ;
10EA :                ; PF1 Print (hard copy) on
10EA : 86 01          PFKY10 LDAA   #$1    ; On print flag
10EC : 20 01                 BRA    PFKY25
10EE :                ; PF2 Print (hard copy) off
10EE : 4F             PFKY20 CLRA          ; Off print flag
10EF : B7 11 E4       PFKY25 STAA   PRTFLG
10F2 : 39             BRKRTN RTS
10F3 :                ;
10F3 :                ; PF3 1200bps
10F3 : CC 3D 47       PFKY30 LDD    #$3D47 ; Set mode (stop: 1, CD: no-check, RTS: on, parity: E,
10F6 :                ;                    ; 7 bits length, 1200bps)
10F6 : FD 11 E8              STD    RSPARM ; Save parameters
10F9 : BD FF 7F       PFKY35 JSR    RSCLOS ; Close RS232C for open again
10FC : FC 11 E8              LDD    RSPARM ; Change bit rate
10FF : BD FF 88              JSR    RSMST
1102 : 38                    PULX
1103 : CE 10 66              LDX    #INIT30; Rewrite return address
1106 : 3C                    PSHX
1107 : 39                    RTS
1108 :                ;
1108 :                ; PF4 300bps
1108 : CC 3D 27       PFKY40 LDD    #$3D27 ; Set mode (stop: 1, CD: no-check, RTS: on, parity: E,
110B :                ;                    ; 7 bits length, 300bps)
110B : FD 11 E8              STD    RSPARM ; Save parameters
110E : 86 01                 LDAA   #1     ; Display monitor = on
```

```
1110 : B7 11 EA                STAA   MONFLG
1113 : 20 E4                   BRA    PFKY35
1115 :                  ; PF6 Monitor on
1115 : 86 01            PFKY60 LDAA   #1
1117 : 20 06                   BRA    PFKY75
1119 :                  ; PF7  Monitor off
1119 : 86 01            PFKY70 LDAA   #1     ; Hard copy = on
111B : B7 11 E4                STAA   PRTFLG
111E : 4F                      CLRA
111F : B7 11 EA         PFKY75 STAA   MONFLG
1122 : 39                      RTS
1123 :                  ; PF8 Esc 'I'+$20 '1'
1123 : 86 19            PFKY80 LDAA   #$19   ; Esc
1125 : 8D 8F                   BSR    PSHCHR
1127 : 86 69                   LDAA   #'I'+$20      ; 'I'+$20
1129 : 8D 8B                   BSR    PSHCHR
112B : 86 31                   LDAA   #'1'   ; '1'
112D : 8D 87                   BSR    PSHCHR
112F : 39                      RTS
1130 :                  ; PF5 Quit
1130 : BD FF 7F         PFKY50 JSR    RSCLOS ; Close RS232C
1133 : 4F                      CLRA          ; Driver off
1134 : BD FF 85                JSR    RSONOF
1137 : 4F                      CLRA
1138 : BD FF 73                JSR    SERONF
113B : FC 11 E5                LDD    SERADR ; Recover interrupt vector
113E : FD 01 0A                STD    SERVCT+1
1141 : 38                      PULX
1142 : 7E FF 25                JMP MENU
1145 :                  ;
1145 : =$1145           INVLKY EQU    *
1145 : 39                      RTS
```

```
1146 :                  ;
1146 :                  ; Serial receive interrupt (receive RS232C) routine
1146 :                  ; Push received data to printer stack and send the character which is
1146 :                  ; in the printer stack
1146 :                  ;
1146 : =$1146           SERINT EQU    *
1146 : B6 11 E4                LDAA   PRTFLG ; Hard copy = "yes"?
1149 : 27 0F                   BEQ    SERI80 ; No, jump to interrupt routine
114B : 96 11                   LDAA   TRCSR  ; Get data
114D : 96 12                   LDAA   SRDR
114F : 84 7F                   ANDA   #$7F   ; Supress bit 7
1151 : 81 7F                   CMPA   #$7F
1153 : 24 03                   BCC    SERI30 ; Ignore 7F - FF
1155 : BD 10 B6                JSR    PSHCHR
1158 :                  ; Hard copy on
1158 : 8D 05            SERI30 BSR    HRDCPY ; Send 3 characters (9ms)
115A :                  ;
115A : FE 11 E5         SERI80 LDX    SERADR
115D : 6E 00                   JMP    0,X
115F :                  ;
115F :                  ; Print to serial printer
115F :                  ;   This routine called only in interrupt
115F :                  ; Register preserve A
115F :                  ;
115F : 36               HRDCPY PSHA
1160 : B6 11 E4                LDAA   PRTFLG        ; Hard copy = "yes"?
1163 : 27 60                   BEQ    HARD80
1165 :                  ; Yes, printing
1165 : 86 03                   LDAA   #3            ; Copy count = 3 (print 3 characters)
1167 : B7 11 E7                STAA   CPYCNT
116A : 7B 40 02                TIM    #$40,PORT1    ; Printer ready?
116D : 26 56                   BNE    HARD80
```

```
116F :                  ; Are there data in the buffer?
116F : FC 11 EF                LDD    BUFCNT
1172 : 27 51                   BEQ    HARD80
1174 :                  ;
1174 : 71 FB 03                AIM    #$FF-4,PORT2  ; Detach slave MCU (select serial)
1177 : 96 11                   LDAA   TRCSR
1179 : 36                      PSHA                 ; Save TRCSR
117A : 86 05                   LDAA   #$05          ; 4800bps
117C : 97 10                   STAA   RMCR
117E : 86 0A                   LDAA   #$0A
1180 : 97 11                   STAA   TRCSR
1182 :                  ;
1182 : FE 11 ED         HARD10 LDX    BPOUT         ; Load data from the stack
1185 : A6 00                   LDAA   0,X
1187 : 08                      INX
1188 : 8C 21 F1                CPX    #BUF+BUFSIZ
118B : 26 03                   BNE    HARD20
118D : CE 11 F1                LDX    #BUF
1190 : FF 11 ED         HARD20 STX    BPOUT         ; Increment data pointer at the buffer
1193 : FE 11 EF                LDX    BUFCNT
1196 : 09                      DEX
1197 : FF 11 EF                STX    BUFCNT
119A :                  ;
119A : 7B 20 11         HARD30 TIM    #$20,TRCSR    ; Wait ready
119D : 27 FB                   BEQ    HARD30
119F : 97 13                   STAA   STDR          ; Store data to the transmit register
11A1 : 7A 11 E7                DEC    CPYCNT        ; Were 3 characters sent?
11A4 : 27 0A                   BEQ    HARD40
11A6 : 7B 40 02                TIM    #$40,PORT1    ; Printer ready?
11A9 : 26 05                   BNE    HARD40
11AB : FC 11 EF                LDD    BUFCNT        ; Is buffer empty?
11AE : 26 D2                   BNE    HARD10
```

```
11B0 :                ;
11B0 :                ; Wait 2ms (time of sending one character)
11B0 : 7B 20 11       HARD40 TIM    #$20,TRCSR
11B3 : 27 FB                 BEQ    HARD40
11B5 : CE 01 90              LDX    #400
11B8 : 09             HARD50 DEX
11B9 : 26 FD                 BNE    HARD50
11BB :                ; Recover serial communication
11BB : 86 04                 LDAA   #$04          ; Select slave MCU
11BD : 97 10                 STAA   RMCR
11BF : 32                    PULA                 ; Recover TRCSR
11C0 : 97 11                 STAA   TRCSR
11C2 : 72 04 03              OIM    #$4,PORT2
11C5 : 32             HARD80 PULA
11C6 : 39                    RTS
11C7 :                ;
11C7 :                ;
11C7 : 84             SCRPKD FCB    $84    ; Screen device select (LCD)
11C8 : 22                    FCB    $22
11C9 :                ;
11C9 : 87                    FCB    $87    ; Set screen size and buffer address
11CA : 13 03                 FCB    19,3
11CC : 21 F1                 FDB    SCRBUF
11CE :                ;
11CE : C3                    FCB    $C3    ; Set cursor margin
11CF : 04                    FCB    4
11D0 :                ;
11D0 : C4                    FCB    $C4    ; Set scroll step
11D1 : 0A                    FCB    10     ; X
11D2 : 03                    FCB    3      ; Y
11D3 :                ;
11D3 : CB                    FCB    $CB    ; Set scroll speed
```

```
11D4 : 09                      FCB    9
11D5 :                  ;
11D5 : =$11D5           SCRPKE EQU    *
11D5 :                  ;
11D5 :                  ;
11D5 :                  ; Work area
11D5 : 84               SCRPK1 FCB    $84    ; Screen device select (LCD)
11D6 : 22                      FCB    $22
11D7 : 87               SCRPK2 FCB    $87    ; Set screen size and buffer address
11D8 : 13 03                   FCB    19,3
11DA : 21 F1                   FDB    SCRBUF
11DC :                  ;
11DC : C3               SCRPK3 FCB    $C3    ; Set cursor margin
11DD : 04                      FCB    4
11DE :                  ;
11DE : C4               SCRPK4 FCB    $C4    ; Set scroll step
11DF : 0A                      FCB    10     ; X
11E0 : 03                      FCB    3      ; Y
11E1 :                  ;
11E1 : CB               SCRPK5 FCB    $CB    ; Set scroll speed
11E2 : 09                      FCB    9
11E3 :                  ;
11E3 : 01               ECHO   FCB    1      ; Terminal mode echo
11E4 :                  ;
11E4 :                  PRTFLG RMB    1      ; Hard copy (MP-80 printer) on/off flag
11E5 :                  ;
11E5 :                  ;
11E5 :                  SERADR RMB    2      ; Save serial receive interrupt vector
11E7 :                  CPYCNT RMB    1      ; Work for hard copy
11E8 :                  RSPARM RMB    2      ; RS232C open parameters
11EA :                  MONFLG RMB    1      ; Display received character = yes?
11EB :                  ;                    ; 1: display; 0: display off
```

```
11EB :                    ;
11EB :                    ; Serial send buffer
11EB :                    BPIN   RMB    2      ; Pointer where next character is stored
11ED :                    BPOUT  RMB    2      ; Pointer where next character is loaded
11EF :                    BUFCNT RMB    2      ; Number of characters in the buffer
11F1 :                    BUF    RMB    BUFSIZ ; Buffer
21F1 :                    ;
21F1 :                    SCRBUF RMB    SCBSIZ ; Screen buffer
22B9 :                    RSBUFF RMB    RSBSIZ ; RS232C receive buffer
32B9 : =$32B9             WRKEND EQU    *
32B9 :                    ;
32B9 :                    ;
32B9 :                           END
```

5.13.4 Control half-duplex modem

```
   0 :                    ; MODEM
   0 :                    ; Control of half duplex modem
   0 :                    ; TSS terminal of half duplex modem
   0 :                    ; Without hard copy
   0 :                    ; By K.A.
   0 :                    ;
   0 :                           PAGE   0
   0 :                           CPU    6301
   0 :                    ; Control half duplex modem
   0 :                    ;
   0 : =$FF19             SNSCOM EQU    $FF19
   0 : =$FF16             CHKRS  EQU    $FF16
   0 :                    ;
   0 :                    ;
   0 :                    ; Constant value
   0 : =$FD               RSPRM1 EQU    $FD    ; Stop bits = 1, carrier detect: check
   0 :                       ; RTS: low, CTS: check, DSR: check, parity: none
   0 : =$48        RSPRM2 EQU    $48    ; Bit length = 8, bit rate = 1200bps
   0 :             ;
   0 :             ;
1000 :                    ORG    $1000
1000 :             ; Entry point of "start RS232C communication" procedure
1000 :             ;   1. RTS low, set bit rate, driver on
1000 :             ;   2. Start to receive
1000 :             ;
1000 :             ; Parameters
1000 :             ;   On entry none
1000 :             ;   On exit none
1000 :             ;
1000 :             ; Subroutine entry point
1000 : =$FF4F      DSPSCR EQU    $FF4F  ; Display one character to virtual screen
1000 : =$FF5E      SCRFNC EQU    $FF5E  ; Virtual screen function
1000 : =$FF85      RSONOF EQU    $FF85  ; RS232C driver on/off
1000 : =$FF88      RSMST  EQU    $FF88  ; Set RS232C parameters
1000 : =$FF82      RSOPEN EQU    $FF82  ; Open RS232C receive
1000 : =$FF7F      RSCLOS EQU    $FF7F  ; Close RS232C receive
1000 : =$FF79      RSGET  EQU    $FF79  ; Get RS232C one character
1000 : =$FF76      RSPUT  EQU    $FF76  ; Send RS232C one character
1000 : =$FF9A      KEYIN  EQU    $FF9A  ; Get one character from keyboard buffer
1000 : =$FF9D      KEYSTS EQU    $FF9D  ; Get number of characters in the key buffer
1000 :             ; Constants or registers
1000 : =$2         PORT1  EQU    $02    ; I/O PORT1
1000 : =$3         PORT2  EQU    $03    ; I/O PORT2
1000 : =$1000      RSBSIZ EQU    4096   ; Buffer size for RS232C
1000 : =$55        SCBSIZ EQU    85     ; Buffer size for screen
1000 : =$1         ECHODT EQU    1      ; Terminal mode = "echo character"?
1000 :             ;                    ; 0: yes; 1: no
1000 :             ;
```

```
1000 :                  ORG    $1000
1000 :           ;
1000 :           ; Initialize
1000 : CE 10 D2         LDX    #SCRPKD       ; Set screen packet X: data address
1003 : C6 0E            LDAB   #SCRPKE-SCRPKD; (B): number of data
1005 : A6 00     INIT10 LDAA   0,X
1007 : A7 0E            STAA   SCRPK1-SCRPKD,X
1009 : 08               INX
100A : 5A               DECB
100B : 26 F8            BNE    INIT10
100D :           ;
100D : CE 10 E0         LDX    #SCRPK1       ; Initialize screen
1010 : BD FF 5E         JSR    SCRFNC        ; Select screen device
1013 : CE 10 E2         LDX    #SCRPK2
1016 : BD FF 5E         JSR    SCRFNC        ; Set screen size and buffer address
1019 : CE 10 E7         LDX    #SCRPK3
101C : BD FF 5E         JSR    SCRFNC        ; Set cursor margin
101F : CE 10 E9         LDX    #SCRPK4
1022 : BD FF 5E         JSR    SCRFNC        ; Set scroll step
1025 : CE 10 EC         LDX    #SCRPK5
1028 : BD FF 5E         JSR    SCRFNC        ; Set scroll speed
102B :           ;
102B : CC 35 47         LDD    #$3547        ; Set mode (stop: 1, CD: no-check, RTS: off,
102E :           ;                           ;          parity: E, 7 bits length, 1200bps)
102E : BD FF 88         JSR    RSMST
1031 : 86 01            LDAA   #1            ; RS232C driver on
1033 : BD FF 85         JSR    RSONOF
1036 : CE 11 51         LDX    #RSBUF        ; (X): buffer address
1039 : CC 10 00         LDD    #RSBSIZ       ; (A,B): buffer size
103C : BD FF 82         JSR    RSOPEN        ; Open RS232C receive
103F :           ;
103F : BD FF 9D  REDKEY JSR    KEYSTS        ; Accept from keyboard?
```

```
1042 : 25 29                   BCS    BRKRTN        ; If Break key pressed, return (in BASIC
1044 :                  ;                            ;   mode)
1044 : 27 14                   BEQ    RCVRS
1046 :                  ; Accepted character from keyboard
1046 : BD FF 9A                JSR    KEYIN
1049 : =$1049           GETKEY EQU    *
1049 : 36                      PSHA
104A : BD FF 4F                JSR    DSPSCR        ; Display character to virtual screen
104D : 32                      PULA
104E : 81 0D                   CMPA   #$0D          ; CR (send data) code?
1050 : 26 08                   BNE    RCVRS
1052 : BD 10 6E                JSR    TXD           ; Transmit data string to RS232C
1055 : 86 0A                   LDAA   #$0A          ; Display 'LF'
1057 : BD FF 4F                JSR    DSPSCR
105A :                  ;
105A : CE FF D8         RCVRS  LDX    #$FFD8        ; Received character from RS232C?
105D : EC 00                   LDD    0,X
105F : 27 DE                   BEQ    REDKEY
1061 : BD FF 79                JSR    RSGET
1064 : 81 7F                   CMPA   #$7F
1066 : 24 D7                   BCC    REDKEY        ; Ignore 7F- FF characters
1068 : BD FF 4F                JSR    DSPSCR
106B : 20 D2                   BRA    REDKEY
106D :                  ;
106D : 39               BRKRTN RTS
106E :                  ;
```

(...missing page in original listings...)

```
10C4 :                  ;
10C4 : 86 4D                   LDAA   #$4D   ; RTS: low
10C6 : BD FF 19                JSR    SNSCOM
```

```
10C9 : 86 00                    LDAA   #$00
10CB : BD FF 19                 JSR    SNSCOM
10CE :                   ;
10CE : BD FF 16                 JSR    CHKRS  ; Restart receiving
10D1 :                   ;
10D1 : 39                       RTS
10D2 :                   ;
10D2 :                   ;
10D2 : 84                SCRPKD FCB    $84    ; Screen device select (LCD)
10D3 : 22                       FCB    $22
10D4 :                   ;
10D4 : 87                       FCB    $87    ; Set screen size and buffer address
10D5 : 13 03                    FCB    19,3
10D7 : 21 51                    FDB    SCRBUF
10D9 :                   ;
10D9 : C3                       FCB    $C3    ; Set cursor margin
10DA : 04                       FCB    4
10DB :                   ;
10DB : C4                       FCB    $C4    ; Set scroll step
10DC : 0A                       FCB    10     ; X
10DD : 03                       FCB    3      ; Y
10DE :                   ;
10DE : CB                       FCB    $CB    ; Set scroll speed
10DF : 09                       FCB    9
10E0 :                   ;
10E0 : =$10E0            SCRPKE EQU    *
10E0 :                   ;
10E0 :                   ;
10E0 :                   ; Work area
10E0 : 84                SCRPK1 FCB    $84    ; Screen device select (LCD)
10E1 : 22                       FCB    $22
10E2 : 87                SCRPK2 FCB    $87    ; Set screen size and buffer address
```

```
10E3 : 13 03                 FCB    19,3
10E5 : 21 51                 FDB    SCRBUF
10E7 :                ;
10E7 : C3             SCRPK3 FCB    $C3    ; Set cursor margin
10E8 : 04                    FCB    4
10E9 :                ;
10E9 : C4             SCRPK4 FCB    $C4    ; Set scroll step
10EA : 0A                    FCB    10     ; X
10EB : 03                    FCB    3      ; Y
10EC :                ;
10EC : CB             SCRPK5 FCB    $CB    ; Set scroll speed
10ED : 09                    FCB    9
10EE :                ;
10EE : 91             SCRPK7 FCB    $91    ; Get extent of virtual screen
10EF :                       RMB    4
10F3 :                ;
10F3 : 97             SCRPK8 FCB    $97
10F4 :                       RMB    90
114E :                ;
114E : 01             ECHO   FCB    1      ; Terminal mode echo
114F :                ;
114F :                PRTFLG RMB    1      ; Hard copy (MP-80 printer) on/off flag
1150 :                ;                    ; 0: off; 1: on
1150 :                TXCNT  RMB    1      ; Number of characters which are sent to host
1151 :                ;                    ;   computer
1151 :                ;
1151 :                RSBUF  RMB    RSBSIZ ; RS232C receive buffer
2151 :                ;
2151 :                SCRBUF RMB    SCBSIZ ; Screen buffer
21A6 :                ;
21A6 :                ;
21A6 :                       END
```

Chapter 6

Cassette input/output

6.1 General

Two types of cassettes may be used as external data storage: an external audio cassette and the built-in microcassette (plug-in option). Data sent to cassettes is recorded sequentially. The average speed of data communication with cassettes is 1300bps. The format of data stored in the external audio cassette and that of the built-in microcassette are the same so the two types of cassettes are compatible. The only control line used for the external audio cassette is the remote ON/OFF line (REM). The built-in microcassette, however, is controlled by software and performs fast forward, rewind, write and playback operations in response to commands in BASIC. The tape counter value is also recorded and displayed by software.

6.2 Data storage (`SAVE`)

1. Format of one bit

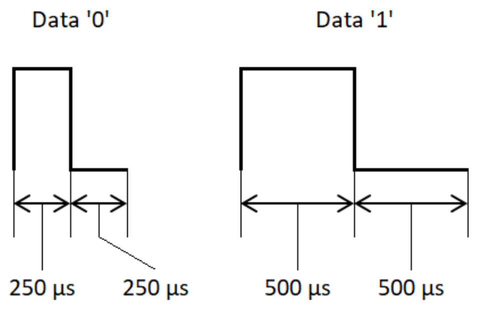


Figure 6.1: Recording format for one bit

In the recording format of the cassette, one bit is represented by one pulse (Figure 6.1).

Each byte, consisting of 8 data bits and one stop bit, is sent from bit 0. The last bit of a byte is the stop bit (data '1'). (Figure 6.2).

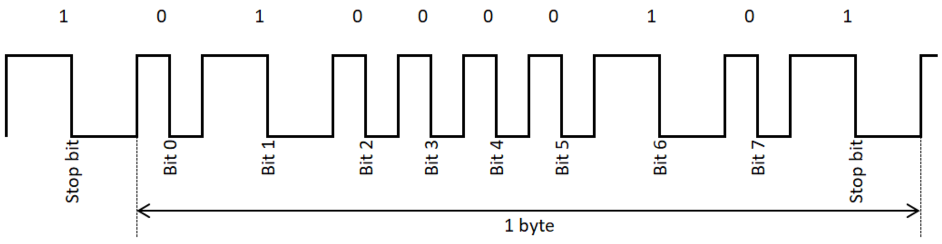


Figure 6.2: Format of one byte

2. Synchronization

   The first bit with data '1' which appears after 40 or more bits of data '0' is taken as the first bit (bit 0) of the synchronization character. Synchronization is performed when the data of this first byte is FF and that of the next byte sent is AA (Figure 6.3).

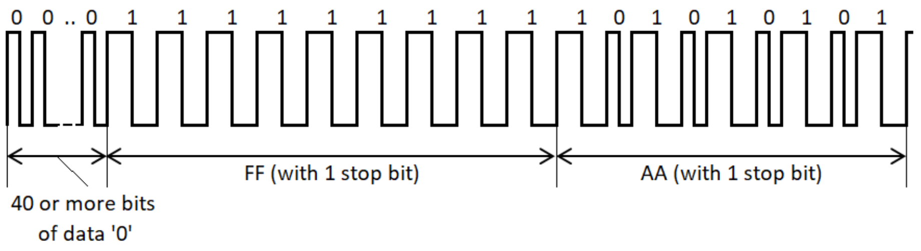


Figure 6.3: Synchronization data

   The next data sent following the synchronization data will be used as actual data.

3. Reverse waveform

   The normal recording format for data bits is as shown in Figure 6.1. However, depending on the cassette used, when the signal passes through the playback circuit of the HX-20, the high/low levels of the waveform may be reversed (Figure 6.4).

   The type of waveform is determined when synchronization is performed and then data read is performed. The waveform of the built-in micro-cassette is inverted.

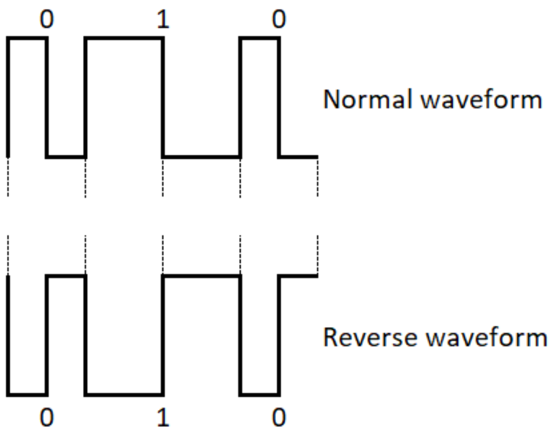


Figure 6.4: Reverse waveform

4. Bit judgement

   To judge whether a bit is ‘`0`’ or ‘`1`’, the interval between the rise of the first pulse and that of the second is measured. If the measured value is above a specified value (approx. 750µs), the bit is judged to be logic ‘`1`’ (Figure 6.5).

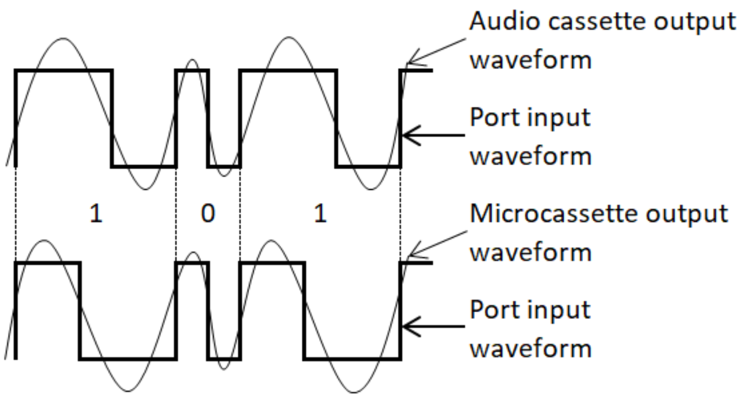


Figure 6.5: Output waveforms

6.3 I/O ports

Table 6.1 lists the I/O ports related to the external cassette.

MCU	Port	Description
Master MCU	P12	Input. Connected to port P34 of the slave MCU, this port informs the master MCU of the slave MCU's error status.
Slave MCU	P30	Output. This port is used for the cassette REM output. 1: off; 0: on.
	P32	Input. This port is used to input data from the external cassette. 1: high; 0: low.
	P33	Output. This port is used to output data to the external cassette. 1: high; 0: low.
	P34	Output. This port is connected to port P12 of the master MCU.

Table 6.1: I/O ports related to the external cassette

Table 6.2 lists the I/O ports related to the microcassette.

MCU	Port	Description
Master MCU	P12	Input. This port is connected to port P34 of the slave MCU and the master MCU of the slave MCU's error status.
	P17	This port is used to input the counter status or to judge the plug-in option.
Slave MCU	P20	Output. This port is used to input data (1: high; 0: low) or to judge the write protection. The handling of the input contents of this port depends on the value of P45.
	P21	Output. This port is used to output data to the microcassette. 1: high; 0: low.
	P42	Output. This port is used to turn the microcassette power on/off. 1: on; 0: off.

Continues in next page...

...continued from previous page

MCU	Port	Description
	`P43`	Output. This port is used to set microcassette commands.
	`P44`	Output. This port supplies a serial clock for timing the microcassette commands. `1`: high; `0`: low.
	`P45`	Output. This port is used to select the `P20` input. `0`: RS-232C; `1`: microcassette.
	`P46`	Input. This port is used to input the counter status when port `P44` is `0` and the head switch status when it is `1`.

Table 6.2: I/O ports related to the microcassette

6.4 Block format

Cassette data is recorded in blocks. Each block consists of the items listed in Table 6.3.

Field	Description
Synchronization field	Contains 80 bits of data '`0`'.
Preamble	Contains data `FF`, `AA` (2 bytes).

Continues in next page...

...continued from previous page

Field	Description
Block identification field	This field consists of 4 bytes. The function of each byte is as follows: • Byte 0: block identifier field indicating the type of block. – `H`: header – `D`: data – `E`: end of file (`EOF`) • Bytes 1 and 2: indicate the 2-byte block number and must be `0000` to `FFFF`. • Byte 3: block identification number. This is used to identify blocks which are written twice to improve reliability. Values `00` through `FF` can be assigned to a block but the values actually used are `00` and `01`.
Data field	Stores data. An 80-byte data field is assigned for header (the block identifier field begins with `H`) and `EOF` blocks (block identifier field begins with `E`). In all other cases, the data field size is defined by the header block.
`BCC` (Block Check Character) field	Performs CRC (Cyclic Redundancy Check) for the range from the beginning of the block to the `BCC` field. The two `BCC` bytes and CRC-CCITT are used for this check.
Postamble	Contains values `AA`, `00` (2 bytes).

Table 6.3: Block configuration

6.5 File structure

Only sequential files are supported. Sequential file data is fixed-length and blocked. Each sequential file consists of an 80-byte header block (the length of the data field excluding the preamble, block identification field, `BCC` and postamble), one or more data blocks (256 bytes each), and an `EOF` block. The block numbers assigned for each file begin with header block `00`, followed by `01`, `02`,... ending with the `EOF` block. Each block is written twice to improve reliability.

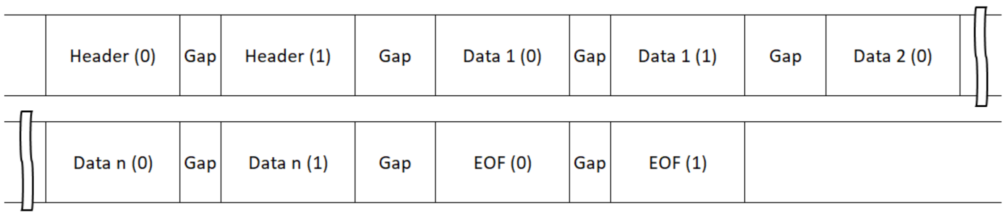


Figure 6.6: Configuration of sequential files

A 5s tape feed (data `FF`) is provided at the beginning and end of each file as a gap to separate files.

6.6 Format of header and `EOF` blocks

The data format of the header block is shown in Table 6.4 and that of the `EOF` block is shown in Table 6.5.

Column from	to	Byte size	Item	Description
0	3	4	ID field	Data `HDR1`. Indicates, in ASCII code, that the block is a header.
4	11	8	Filename	Stores the filename.
12	19	8	File type	Stores the file type.
20	20	1	Record type	This byte specifies the record type. The following record types can be specified. • `F`: fixed length. • `V`: variable length. • `2`: each fixed-length block is written twice HX-20 currently supports only record type `2`.
Continues in next page...				

...continued from previous page				
Column from	to	Byte size	Item	Description
21	21	1	Interblock gap length	This byte specifies the interblock gap length. • "Δ": interblock gap long enough for the tape to stop (long gap). • "S": interblock gap length not long enough for the tape to stop (short gap).
22	26	5	Block length	Indicates the data length of the block. Must be `00000` to `65535` (ASCII code).
27	31	5		Empty.
32	37	6	Creation date	Indicates the date of file creation in "month, day, year" format (ASCII code). Month, day and year are represented by 2 bytes of data each.
38	43	6	Creation time	Indicates the time of file creation in "hour, minutes, seconds" format (ASCII code). Hour, minute and second are represented by 2 bytes of data each. Hour is the indicated by the 24-hour system (`00` to `23`).
44	49	6		Empty.
50	51	2	Volume serial number	Indicates the tape volume number in ASCII code (`01` ∼).
52	59	8	System name	Indicates the name of the system that created the file (ASCII code). "`HX-20ΔΔΔ`".
60	79	20		Empty.

Table 6.4: Format of header block

Column from	to	Byte size	Item	Description
0	3	4	ID field	“EOFΔ”.
4	79	66		Empty.

Table 6.5: Format of EOF block

6.7 Interblock gaps

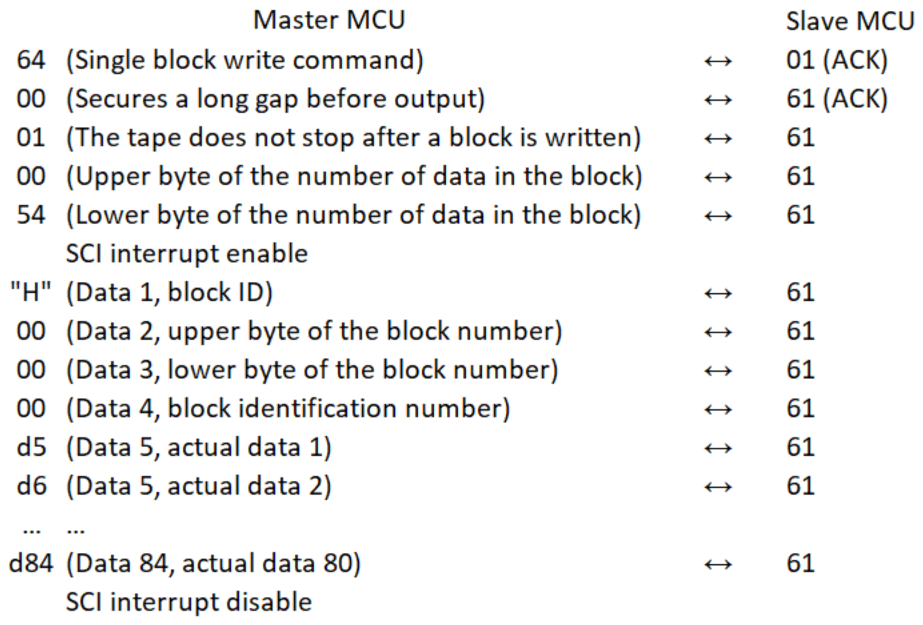
There are two types of interblock gaps: long and short. The length of an interblock gap depends on whether the tape will stop at the gap. An interblock gap of approx. 10 bytes (the length of tape required to write a single block twice) is secured between blocks where the tape will not stop. This is a short gap. An interblock gap of approx. 100 bytes is required when the tape stops between blocks. This type of gap (long gap) enables the motor of the tape drive to reach a constant rotation speed from a halt state. The length of the interblock gap is specified by the header.

6.8 Writing blocks

Data is written to cassettes by the slave MCU in units of one block. Commands for block write are exchanged between the master and the slave MCUs as shown in Figure 6.7. The master MCU must send the write data within 4ms after receiving `ACK` from the slave. The tape drive must already be running when data is sent to the slave MCU. The data cosists of the block ID (“`H`”) and the contents of the data block (84 bytes for the header). CRC calculations are performed solely by the slave MCU.

When the `RIE` (receive interrupt enable) mask of the SCI (serial communication interface) is opened, the main MCU uses the interrupt routine to transmit the data from “`H`” to `d84` in Figure 6.7 to the slave MCU. When master MCU received data 61 from the slave MCU, an `SCI` interrupt is generated and the master MCU sends next data to the slave MCU.

The `RIE` mask is closed after one block has been transmitted. The master MCU can transmit data to the slave MCU without generating an `SCI` interrupt but the current transmission procedure uses the `SCI` interrupt.

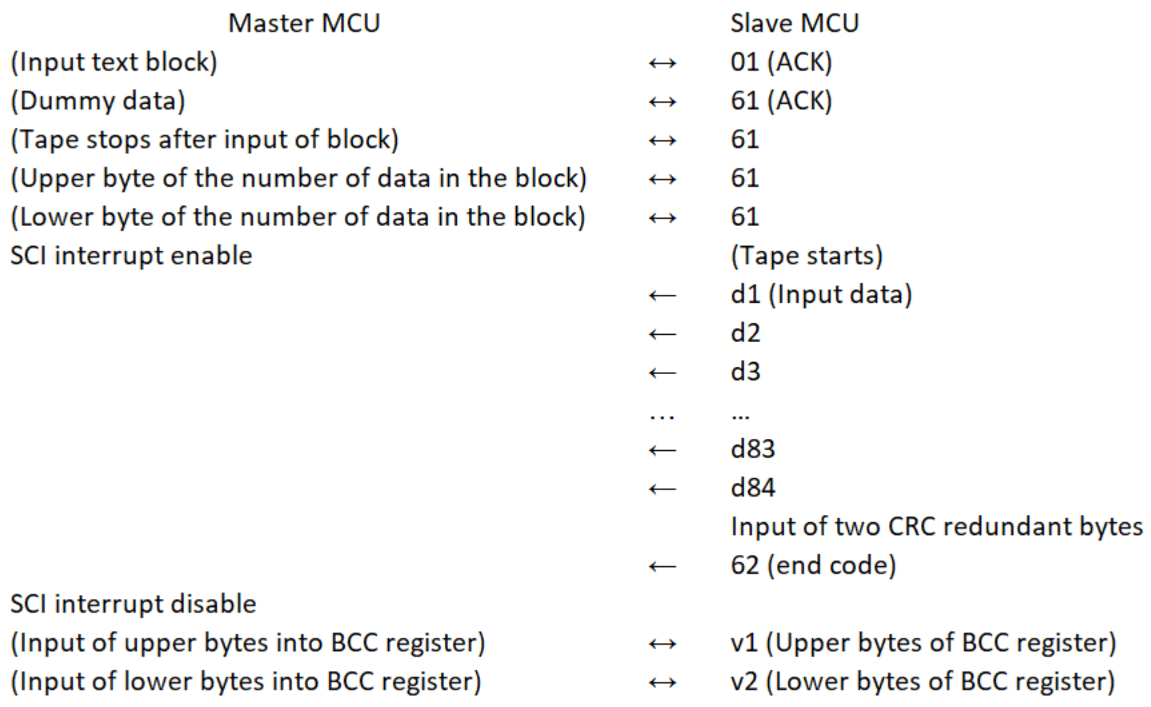


Master MCU			Slave MCU
64	(Single block write command)	↔	01 (ACK)
00	(Secures a long gap before output)	↔	61 (ACK)
01	(The tape does not stop after a block is written)	↔	61
00	(Upper byte of the number of data in the block)	↔	61
54	(Lower byte of the number of data in the block)	↔	61
	SCI interrupt enable		
"H"	(Data 1, block ID)	↔	61
00	(Data 2, upper byte of the block number)	↔	61
00	(Data 3, lower byte of the block number)	↔	61
00	(Data 4, block identification number)	↔	61
d5	(Data 5, actual data 1)	↔	61
d6	(Data 5, actual data 2)	↔	61
...	...		
d84	(Data 84, actual data 80)	↔	61
	SCI interrupt disable		

Figure 6.7: Exchange of commands for write operation for a single block (header)

6.9 Reading blocks

Command `28` (`26`, `27`) is used to read a block from the external casssette. Command `68` (`66`, `67`) is used to read a block from the microcassette. The slave MCU transmits to the master MCU the contents of the block, from the beginning of the block identification field to the beginning of the `BCC`. Redundant bytes used for the CRC check are not sent to the main MCU. When one block has been sent, the slave MCU sends a completion code (`22` for the external cassette and `62` for the built-in microcassette) to the master MCU. When the master MCU receives the completion code, it inputs a `BCC` value to the slave MCU and evaluates the CRC check. CRC check is performed for the range from the block identification field to the CRC redundant byte. If the result of the CRC check is 0, this indicates that the data write operation has been correctly performed. Next, the block number is checked. If block 4 is input when block 5 should be input, the next block must be input. If 6 is input, this means that the desired block has already passed. When a single block has been correctly input, this is taken as the completion of input processing. Otherwise, input processing is aborted or the input procedure for the next block is begun. The master MCU receives the data sent from the slave MCU via the SCI using `SCI` receive interrupt processing and stores this data in the specified buffer.



Master MCU		Slave MCU
(Input text block)	↔	01 (ACK)
(Dummy data)	↔	61 (ACK)
(Tape stops after input of block)	↔	61
(Upper byte of the number of data in the block)	↔	61
(Lower byte of the number of data in the block)	↔	61
SCI interrupt enable		(Tape starts)
	←	d1 (Input data)
	←	d2
	←	d3
	…	…
	←	d83
	←	d84
		Input of two CRC redundant bytes
	←	62 (end code)
SCI interrupt disable		
(Input of upper bytes into BCC register)	↔	v1 (Upper bytes of BCC register)
(Input of lower bytes into BCC register)	↔	v2 (Lower bytes of BCC register)

Figure 6.8: Exchange of commands for read operation for a single block (header)

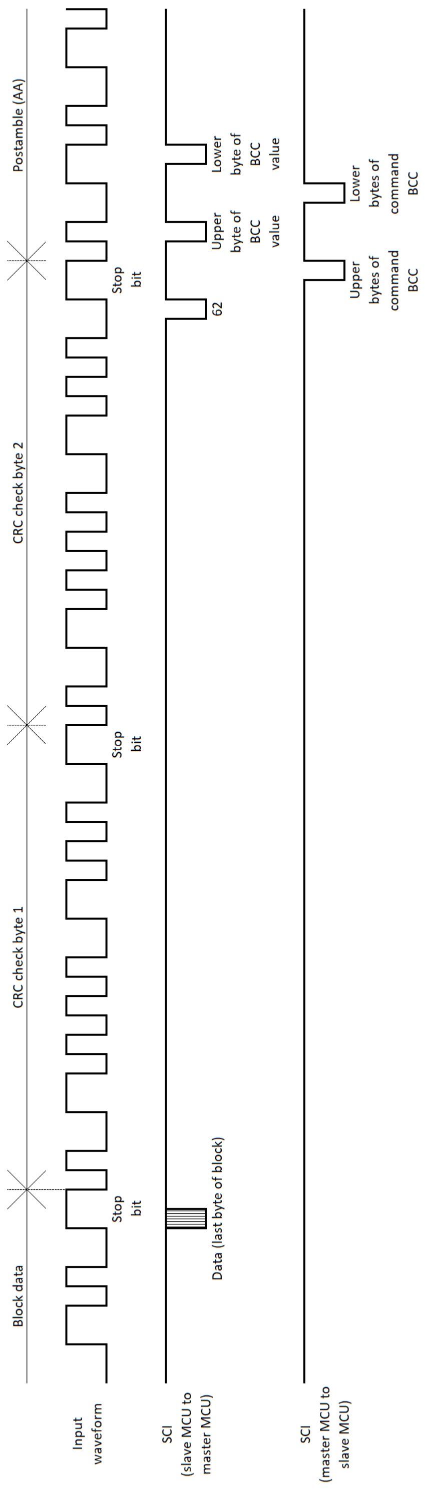


Figure 6.9: Input timing for block data

6.10 File output

Files are output to cassettes using the following three procedures

1. File open

   Subroutine OPNWMS is used to open files for output to the built-in microcassette and subroutine OPNWCS performs the same function for the external cassette. When a file is opened for output, the header is output and internal preparations are made for data block output. Specification for the tape to stop after the head block has been output is included.

2. Output of one byte to a tape file

   Subroutine WRITMS outputs data to the built-in microcassette and subroutine WRITCS to the external cassette. Data is written to a buffer (256 bytes of data + block identification data). Actual output to the microcassette is performed when the buffer becomes full.

3. File close

   Subroutine CLSMS closes the built-in microcassette file and subroutine CLSCS closes the external cassette file. If any data remains in the buffer when the file is closed, it is output as a data block. An EOF block is then output and the tape stops.

6.10.1 Double write

As a measure to improve reliability, the contents of the buffer are output twice (each block is written twice). This procedure is followed for all blocks (header, data and EOF).

6.11 File open

Files are input from cassettes using the following three procedures.

1. File open

   Subroutines SRCRCS and OPNRCS are used to open files for input from the external cassette and subroutines SRCRMS and OPNRMS perform the same function for the built-in microcassette. These files search a specified file by inputting a header from the tape and comparing this with the specified file. After the header of the specified file has been input the tape stops and datainput is internally prepared.

2. Input of one byte from a tape file

   Subroutine READMS inputs one byte of data from the microcassette and subroutine READCS from the external cassette. Data is fetched one byte at a time from the 256-byte buffer. When the buffer is empty, the next block is written to it from the tape and data fetch continues.

3. File close

   Subroutine CLSMS closes the microcassette file and subroutine CLSCS the external cassette file. The tape stops when one of these subroutines is called. When a file is closed, the corresponding input device is released.

6.12 Functions unique to the built-in microcassette

Fast forward and rewind of the microcassette are performed by the slave MCU in response to commands sent from the master MCU. The slave MCU also starts and stops the motor and reads the tape counter value. The following 4 subroutines are provided.

1. MCSMAN: performs the operations in the manual operation mode.
2. REWMCS: rewinds the tape to the beginning.
3. SEKMCS: winds the tape to the specified counter value.
4. CNTMCS: sets or reads the microcassette tape counter value.

6.12.1 Counter read

The main MCU controls the counter during data input or output. The slave MCU controls the counter at all other times. If there is no change in the counter signal for a specified length of time, it is judged that either no tape is set in the drive or that the tape has been wound to the BOT or EOT position. The tape then stops. Port P17 of the main MCU is used to input the tape counter status. This port value indicates whether the tape counter signal is high or low. The tape counter value is indicated by number of changes in the tape counter signal. When data is being input or output, the main MCU inputs the tape counter signal and performs sampling using a TOF interrupt (0.1s interval). The slave MCU controls the counter when fast forward or rewind id being performed.

6.13 Notes on I/O

1. Polynomials generated for CRC check

   The default value ($x^{16} + x^{12} + x^5 + 1$) for polynomial expressions generated for CRC check is set by the slave MCU after reset. This value can be modified by using slave MCU command `48`. If the polynomial expression generated at the time of input is different from that generated at the time of output, the system assumes that a CRC error has occurred and no data can be input.

2. Interblock gaps

   When the REM terminal is used for data output to an external cassette, data write will not be correctly performed if the tape drive takes too much time to reach constant running speed from a fully stopped state. When using a tape recorder where such a condition occurs, the interblock gap must be lengthened (slave MCU command `21`).

3. Number of input data in a block

   When one of the slave MCU commands `28` or `68` (input one block) is input, if the first data input is `H` or `E` (header or `EOF` block), 84 bytes is assumed as the length of the data field of the block and the number of data specified by the command is ignored.

6.14 External cassette subroutines

Subroutine name	Entry point	Description
`PONFCS`	FF46	Turns on/off the remote (REM) terminal.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters – At entry ∗ (A): · 0: turns the REM terminal off. · 1: turns the REM terminal on. Bit 0 is used. – At return ∗ (C): abnormal I/O flag ∗ (A): return codes · 00: normal (00 is always set in the current version). • Registers retained: (B) and (X). • Subroutines referenced – SNSCOM – CHKRS • Variables used: none
OPNRCS	FF43	Opens the cassette file for input and searches the specified file until it is found.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters – At entry ∗ `(X)`: starting address of a data packet Data packet 1. Interblock stop mode (1 byte) · `00`: tape stops at theinterblock gap. · `01`: tape does not stop at the interblock gap. · `FF`: according to the header specification. 2. Starting address of input buffer (two bytes, high- to low-byte order). Input buffer size is 260 bytes. 3. 8-byte filename (ASCII code). 4. 8-byte file type (ASCII code). **Note:** if '`*`' is specified in the character string of a data packet filename, matching terminates at this asterisk position. '`*`' can also be used in a file type. A file whose filename and type match the specified filename and type is assumed to be the specified file. For example, if the filename is "`FILE`" and any file type is acceptable, the filename should be specified as "`FILEΔΔΔΔ`" and the file type as "`*ΔΔ`". To specify the first file in the tape, both filename and type should be "`*ΔΔΔΔΔΔΔ`". – At return ∗ `(C)`: abnormal I/O flag ∗ `(A)`: return codes · `00`: normal. · `85`: file error. ∗ `(Z)`: according to value of `(A)`.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Registers retained: none. • Subroutines referenced – `SNSCOM` – `CRDBHD` – `CRDBEF` • Variables used: `R1`, `R2`, `R3`, `R4` and `R5`.
`SRCRCS`	`FF40`	Opens the cassette file when the first file found is the specified file. Returns the found filename.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters – At entry ∗ `(X)`: top address of data packet. Data packet 1. Interblock stop mode: same as for subroutine `OPNRCS`. 2. Starting address of input buffer: same as for subroutine `OPNRCS`. 3. Filename: same as for subroutine `OPNRCS`. 4. File type: same as for subroutine `OPNRCS`. 5. Found filename (8 bytes). 6. Found file type (8 bytes). **Note:** the function of "`*`" in the specification of filename and type is the same as for subroutine `OPNRCS`. – At return ∗ `(C)`: abnormal I/O flag ∗ `(A)`: return codes · `00`: normal. · `85`: file error. · `8B`: file found is not the specified file. ∗ `(Z)`: according to value of `(A)`. • Registers retained: none. • Subroutines referenced – `SNSCOM` – `CRDBHD` – `CRDBEF` • Variables used: `R0`, `R1`, `R2`, `R3`, `R4` and `R5`.

Continues in next page...

...continued from previous page.		
Subroutine name	Entry point	Description
`READCS`	`FF3D`	Inputs one byte of data from the external cassette. Input data is fetched from the 256-byte buffer one byte at a time. When the buffer becomes empty, the next block is automatically written to the buffer.
		• Parameters – At entry: none – At return ∗ `(C)`: abnormal I/O flag ∗ `(A)`: 1-byte input data. ∗ `(B)`: return codes · `00`: normal. · `01`: end of file (`EOF`). · `84`: the input file is not open. · `81`: read error ∗ `(Z)`: according to the value of `(B)`. • Registers retained: `(X)`. • Subroutines referenced: `CRDBLK`. • Variables used: `R0`, `R1`, `R2`, `R3`, `R4` and `R5`.
`OPNWCS`	`FF3A`	Opens the external cassette file for output.
Continues in next page...		

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters – At entry ∗ `(X)`: top address of data packet. Data packet 1. Interblock stop mode (1 byte) · `00`: tape stops at the interblock gap. · `01`: tape does not stop at the interblock gap. 2. Starting address of input buffer (buffer size is 260 bytes). 3. 8-byte filename (ASCII code). 4. 8-byte file type (ASCII code). – At return ∗ `(C)`: abnormal I/O flag ∗ `(A)`: return codes · `00`: normal. · `88`: file is already open. · `91`: output error. ∗ `(Z)`: according to value of `(A)`. • Registers retained: none. • Subroutines referenced: `CWRHED` • Variables used: `R0`, `R1`, `R2`, `R3`, `R4` and `R5`.
`WRITCS`	`FF37`	Outputs one byte of data to the external cassette. Output data is written to the 260-byte buffer. When the buffer becomes full, data is automatically written to the file.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters – At entry: ∗ `(A)`: 1-byte output data. – At return ∗ `(C)`: abnormal I/O flag ∗ `(B)`: return codes · `00`: normal. · `94`: file is not open. · `91`: output error. ∗ `(Z)`: according to the value of `(B)`. • Registers retained: `(A)` and `(X)`. • Subroutines referenced: `CWRBLK`. • Variables used: `R0`, `R1`, `R2`, `R3`, `R4` and `R5`.
`CLSCS`	`FF34`	Closes the external cassette file. When an output file is closed, any data remaining in the buffer is output to the cassette followed by an `EOF` block. When an input file is closed, input operation simply terminates.

Continues in next page...

...continued from previous page.		
Subroutine name	Entry point	Description
		• Parameters – At entry: none. – At return ∗ `(C)`: abnormal I/O flag ∗ `(A)`: return codes · `00`: normal. · `87`: file is not open. · `91`: output error. ∗ `(Z)`: according to the value of `(A)`. • Registers retained: none. • Subroutines referenced: – `WRTCCS`. – `CWRHED`. – `SNSCOM`. • Variables used: `R0`, `R1`, `R2`, `R3`, `R4` and `R5`.

6.15 Built-in microcassette subroutines

Subroutine name	Entry point	Description
`MCSMAN`	`FF0D`	Performs FF (fast forward) and REW (rewind), etc., according to the keyboard input and displays the tape counter value on the LCD. The keys used for the manual operation mode as follows. • PF1: FF. • PF2: slow forward. • PF3: stop. • PF4: REW. • PF5: quit. Returns from the subroutine. • PF6: counter reset. This subroutine preserves the contents of the virtual screen while the HX-20 is in the manual operation mode.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters – At entry: none. – At return ∗ `(C)`: abnormal I/O flag ∗ `(A)`: return codes · `00`: normal. · `80`: microcassette is not mounted. ∗ `(Z)`: according to the value of `(A)`. • Registers retained: none. • Subroutines referenced: – `KEYIN`. – `KEYSTS`. – `SNSCOM`. – `DSPLCN`. – `BINDEC`. – `LRECV`. • Variables used: none.
`OPNRMS`	`FF0A`	Opens the microcassette file for input and searched the specified file until it is found (see subroutine `OPNRCS`).

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters: at entry and return, same as subroutine OPNRCS except that return code 80 is also used. • Registers retained: same as subroutine OPNRCS. • Subroutines referenced: MWRHED. • Variables used: same as subroutine OPNRCS.
SRCRMS	FF07	Opens the microcassette file. The function of this subroutine is the same as that of subroutine SRCRCS.
		• Parameters: at entry and return, same as subroutine SRCRCS except that return code 80 is also used. • Registers retained: same as subroutine SNSCOS. • Subroutines referenced – SNSCOM – MRDBHD – MRDBEF • Variables used: same as subroutine SNSCOS.
READMS	FF04	Inputs one byte of data from the microcassette. The function of this subroutine is the same as that of subroutine READCS.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters: at entry and return, same as subroutine `READCS` except that return code `80` is also used. • Registers retained: same as subroutine `READCS`. • Subroutines referenced – `SNSCOM` – `WRTMCS` – `MWRHED` • Variables used: same as subroutine `READCS`.
`OPNWMS`	`FF01`	Opens the microcassette file.
		• Parameters: at entry and return, same as subroutine `OPNWCS` except that return code `80` is also used. • Registers retained: same as subroutine `OPNWCS`. • Subroutines referenced: `MWRHED` • Variables used: same as subroutine `OPNWCS`.
`WRITMS`	`FEFE`	Outputs one byte of data to the microcassette. The function of this subroutine is the same as that of subroutine `WRITCS`.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters: at entry and return, same as subroutine WRITCS except that return code 80 is also used. • Registers retained: same as subroutine WRITCS. • Subroutines referenced: MWRBLK • Variables used: same as subroutine WRITCS.
CLSMS	FEFB	Closes the microcassette file. The function of this subroutine is the same as that of subroutine CLSCS.
		• Parameters: at entry and return, same as subroutine CLSCS except that return code 80 is also used. • Registers retained: same as subroutine CLSCS. • Subroutines referenced – SNSCOM – WRTMCS – MWRHED • Variables used: same as subroutine CLSCS.
REWMCS	FEF5	Rewinds the microcassette tape to the beginning.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters – At entry: none. – At return ∗ `(C)`: abnormal I/O flag ∗ `(A)`: return codes · `00`: normal. · `80`: microcassette is not mounted. ∗ `(X)`: tape counter value after rewind (-32768 to 32767). ∗ `(Z)`: according to the value of `(A)`. • Registers retained: none. • Subroutines referenced: – `CHKMCS` – `SNSCOM` • Variables used: `R0`.
`SEKMCS`	`FEF2`	Winds the microcassette tape to the specified tape counter value.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Description
		• Parameters – At entry ∗ `(X)`: specified value of the binary counter (-32768 through 32767). – At return ∗ `(C)`: abnormal I/O flag ∗ `(A)`: return codes · `00`: normal. · `80`: microcassette is not mounted. ∗ `(X)`: counter value after wind. ∗ `(Z)`: according to the value of `(A)`. • Registers retained: none. • Subroutines referenced: – `CHKMCS` – `SNSCOM` • Variables used: `R0`.
`CNTMCS`	`FEEF`	Sets or reads the microcassette tape counter value.

Continues in next page...

…continued from previous page.

Subroutine name	Entry point	Description
		• Parameters – At entry ∗ `(A)`: specifies setting or reading of the tape counter value. · `00`: reads the tape counter value. · `01`: sets the tape counter value (any value other than `00` is taken as `01`). ∗ `(X)`: counter value (`A` ≠ `00`). – At return ∗ `(C)`: abnormal I/O flag (`0` is always set on return). ∗ `(X)`: counter value (`A` = `00` at entry). • Registers retained: `(B)`. • Subroutines referenced: none. • Variables used: none.

6.16 Work areas for external cassette

Address (from)	Address (to)	Variable name	Byte count	Description
1D5	1D5	CSMOD	1	Current mode • Bits 1 and 0: format – 00: EPSON format. – Other than 00: format other than EPSON format. • Bits 3 and 2: file open status – 00: file not open. – 01: open for input. – 10: open for output. – 11: undefined. • Bits 4 to 7: undefined.
1D6	1D7	CSBLNO	2	Block number.
1D8	1D9	CSBCC	2	BCC register value (CRC check for a single block).
1DA	1DB	CSBLSZ	2	Unused.
1DC	1DC	CSBSTP	1	Interblock gap tape stop mode. • 0: tape stops at the interblock gap. • 1: tape does not stop at the interblock gap.

Continues in next page...

...continued from previous page.				
Address (from)	(to)	Variable name	Byte count	Description
`1DD`	`1DD`	`CSSTS`	1	Error status (logic '`1`' in any bit indicates an error). • Bit 0: `EOF` (`EOF` detected during input). • Bits 1 to 4: undefined. • Bit 5: write error. • Bit 6: read error. • Bit 7: buffer overflow.
`1DE`	`1DF`	`CSBFAD`	2	Starting address of cassette buffer.
`1E0`	`1E1`	`CSBFBT`	2	Ending address of cassette buffer plus 1.
`1E2`	`1E3`	`CSBFSZ`	2	Cassette buffer size (in bytes).
`1E4`	`1E5`	`CSBFIP`	2	Pointer indicating the next address to be stored in the cassette buffer.
`1E6`	`1E7`	`CSBFOP`	2	Pointer indicating the next address to be fetched from the cassette buffer.
`1E8`	`1E9`	`CSBFCM`	2	Number of data in buffer.
`1EA`	`1EA`	`CSRDTR`	1	Upper limit for the number of block input trials.
`1EB`	`1EB`	`CSRDCN`	1	Number of block input trials.

6.17 Work areas for built-in microcassette

Address (from)	Address (to)	Variable name	Byte count	Description
1EC	1EC	MSMOD	1	Current mode • Bits 1 and 0: format – 00: EPSON format. – Other than 00: format other than EPSON format. • Bits 3 and 2: file open status – 00: file not open. – 01: open for input. – 10: open for output. – 11: undefined. • Bits 4 to 7: undefined.
1ED	1EE	MSBLNO	2	Block number.
1EF	1F0	MSBCC	2	BCC register value (CRC check for a single block).
1F1	1F2	MSBLSZ	2	Unused.
1F3	1F3	MSBSTP	1	Interblock gap tape stop mode. • 0: tape stops at the interblock gap. • 1: tape does not stop at the interblock gap.

Continues in next page...

...continued from previous page.

Address (from)	Address (to)	Variable name	Byte count	Description
1F4	1F4	CSSTS	1	Error status (logic '1' in any bit indicates an error). • Bit 0: EOF (EOF detected during input). • Bits 1 to 3: undefined. • Bit 4: counter not updated. • Bit 5: write error. • Bit 6: read error. • Bit 7: buffer overflow.
1F5	1F6	MSBFAD	2	Starting address of microcassette buffer.
1F7	1F8	MSBFBT	2	Ending address of microcassette buffer plus 1.
1F9	1FA	MSBFSZ	2	Microcassette buffer size (in bytes).
1FB	1FC	MSBFIP	2	Pointer indicating the next address to be stored in the buffer.
1FD	1FE	MSBFOP	2	Pointer indicating the next address to be fetched from the buffer.
1FF	200	MSBFCM	2	Number of data in buffer.
201	201	MSRDTR	1	Upper limit for the number of block input trials.
202	202	MSRDCN	1	Number of block input trials.
203	204	MSCNTR	2	Counter value.
205	205	MSMNCM	1	Manual command currently being executed.
206	206	MTOFCN	1	Sampling timeout counter for data I/O.
207	207	MSPLMD	1	Counter pulse status (low or high).

6.18 Work areas for external cassette headers

Address (from)	(to)	Variable name	Byte count	Description
2D0	2D0	CHBLID	1	‘H’
2D1	2D2	CHBLNO	2	Block number (binary, 0...)
2D3	2D3	CHBLBU	1	Same block, block number (0, 1...)
2D4	2D7	CID	4	‘HDR’
2D8	2DF	CFNAME	8	Filename.
2E0	2E7	CFTYPE	8	File type.
2E8	2E8	CRTYPE	1	Record type (2: double write).
2E9	2E9	CBMODE	1	Block mode • S: short gap. • Δ: interblock gap stop.
2EA	2EE	CBLNG	5	Block length (ΔΔ256: 256).
2EF	2F3		5	
2F4	2F9	CDATE	6	Date (MMDDYY).
2FA	2FF	CTIME	6	Time (HHMMSS).
300	305		6	
306	307	CVOLN	2	Volume number.
308	30F	CSYSN	8	System name (HX-20ΔΔΔ).
310	323		20	

6.19 Work areas for built-in microcassette headers

Address (from)	(to)	Variable name	Byte count	Description
324	324	MHBLID	1	‘H’
325	326	MHBLNO	2	Block number.
327	327	MHBLBU	1	Same block, block number.
328	32B	MID	4	‘HDR1’
32C	333	MFNAME	8	Filename.
334	33B	CFTYPE	8	File type.
33C	33C	MRTYPE	1	Record type (2: double write).

Continues in next page...

...continued from previous page.				
Address (from)	(to)	Variable name	Byte count	Description
33D	33D	MBMODE	1	Block mode • S: short gap. • Δ: interblock gap stop.
33E	342	MBLNG	5	Block length (ΔΔ256: 256).
343	347		5	
348	34D	MDATE	6	Date (MMDDYY).
34E	353	MTIME	6	Time (HHMMSS).
354	359		6	
35A	35B	MVOLN	2	Volume number.
35C	363	MSYSN	8	System name (HX-20ΔΔΔ).
364	377		20	
378	47B	CASBUF	260	Buffer used by the microcassette.

Chapter 7

Microprinter

7.1 General

The built-in microprinter is a dot matrix printer with a print width of 144 dots. Printing is performed by a single print head driven by four solenoids. Print mode is unidirectional and paper feed is performed each time the print head is returned. The I/O ports related to printing are connected to the slave MCU which controls printing. The bit patterns for printing, however, are supplied by the master MCU.

7.2 Print heads and solenoids

The microprinter has one print head and four solenoids: A, B, C and D. Each solenoid prints 36 dots during a single pass of the print head (Figure 7.1).

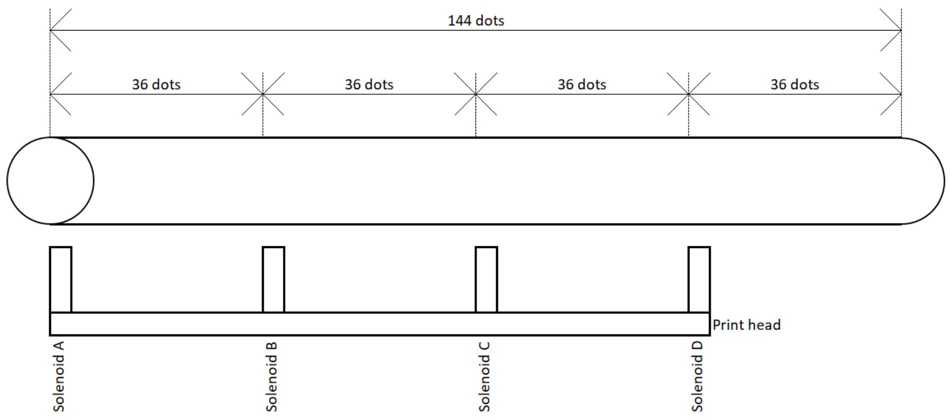


Figure 7.1: Print area of each solenoid

Only unidirectional printing is performed and line feed of one dot-line is performed when the head is returned (Figure 7.2).

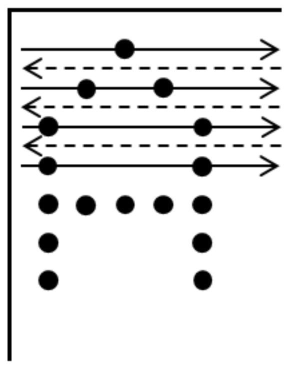


Figure 7.2: Print head operation

Thus, to print a single 6 × 8-dot character pattern, the print head must make 8 passes in each direction.

When printing "`ABCDEFGHIJKLMNOPQRSTUVWX`", characters "`ABCDEF`" are printed by solenoid A, "`GHIJKL`" are printed by solenoid B, "`MNOPQR`" by solenoid C, and "`STUVWX`" by solenoid D.

The printer is controlled by the slave MCU, but actual printing is performed in response to commands sent from the master MCU.

7.3 Ports

The I/O ports related to the printer are as follows.

MCU	Port	Input/ Output	Function
Slave MCU	`P10`	Output	Print solenoid 1 • `1`: ON. • `0`: OFF.
	`P11`	Output	Print solenoid 2 • `1`: ON. • `0`: OFF.
Continues in next page...			

...continued from previous page			
MCU	Port	Input/ Output	Function
	`P12`	Output	Print solenoid 3 • `1`: ON. • `0`: OFF.
	`P13`	Output	Print solenoid 4 • `1`: ON. • `0`: OFF.
	`P14`	Output	Motor output • `1`: ON. • `0`: OFF.
	`P15`	Input	Reset signal input • `1`: high. • `0`: low.
	`P16`	Input	Timing pulse • `1`: high. • `0`: low.
	`P17`	Output	Motor break • `1`: break ON. • `0`: break OFF.

Note: commands must not be sent from the master MCU which will operate the above ports to supply current to the print solenoids for more than a few seconds or to supply a `BREAK` signal while motor output is specified (`P14` is `1`).

7.4 Slave MCU commands

The slave MCU is provided with a command for printing 6 dots of print data. This command is sent from the master MCU 24 times to print one dot-line. Therefore, sending this command 48 times will print 2 dot-lines and sending it 192 (24×8) times will print one line of 6×8-dot character patterns.

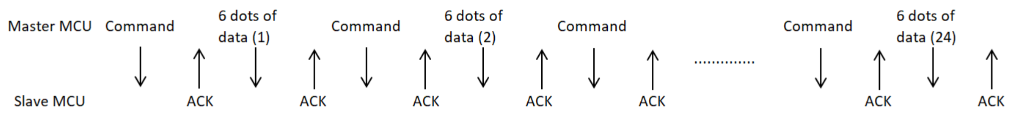


Figure 7.3: Transmission of slave MCU command

If printing is resumed after being interrupted (the print head stops), a blank of one dot-line will occur. This is due to the automatic paper feed (one pitch) when the print head is returned and to the fact that the head stop and restart operation has not finished within the duration of the head's return pass across the page.

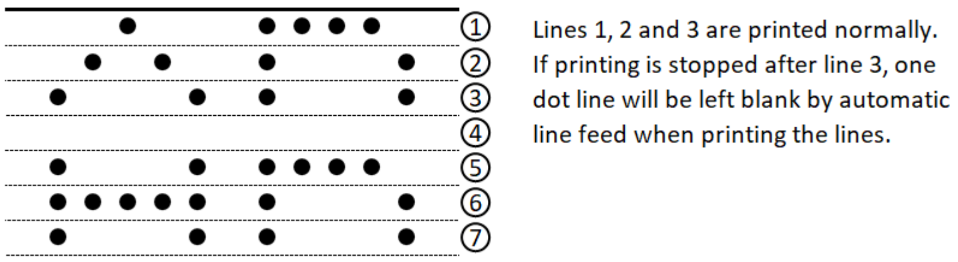

①
②
③
④
⑤
⑥
⑦
Lines 1, 2 and 3 are printed normally.
If printing is stopped after line 3, one dot line will be left blank by automatic line feed when printing the lines.

Figure 7.4: One blank dot line when print head stops

After the slave MCU restarts printing on the printer and a new line is to be printed, if there are less than 24 bytes of data in the data buffer, printing is stopped automatically. When continuously printing a given print pattern, if an interrupt in command transmission from the master to the slave MCU of approx. 300ms occurs, data may be lost (Figure 7.5).

In Figure 7.5, printing of an `A` pattern has been attempted. After the data on line 4 has been sent to the slave MCU and blank time has passed, data transmission is performed. Since there is only one byte of data in the

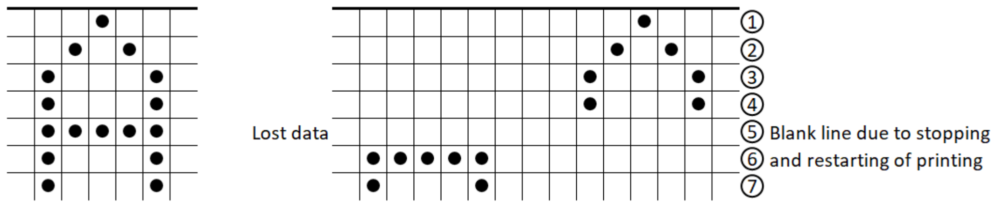


Figure 7.5: Loss of print data

slave MCU, printing is stopped. The data in the buffer at this time will be lost.

Printing is resumed when the contents of the buffer exceed 24 bytes. This results in lost print data, as shown in Figure 7.5.

When printing a line of characters (subroutine `LNPRNT`), after 8 dot-lines of data have been sent, a 2-dot line feed command is sent from the master MCU. In this way, data loss due to timing is prevented (since the feed command processed by the slave MCU stops fetching of the dot pattern data to the buffer).

7.5 List of printer subroutines

Subroutine name	Entry point	Contents
`CHPRNT`	`FF97`	Outputs one character to the microprinter. All control codes (`00-1F`) except `CR` (`0D`) and `LF` (`0A`) are ignored. For `CR`, the buffer column position is set to 0 (first position) and the contents of the buffer are cleared. For `LF` control codes, the contents of the buffer are printed. After printing, the buffer is cleared and the buffer column position is set to 0.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
		• Parameters – At entry ∗ `(A)`: character code (ASCII). – At return ∗ `(C)`: abnormal I/O flag • Registers retained: `(A)`, `(B)`, `(X)`. • Subroutines referenced: – `LNPRNT`. – `CLRB`. • Variables used: none.
`LNPRNT`	`FF94`	Outputs one line of characters to the microprinter. Checks for printer switch ON or OFF. If OFF, the output procedure is ignored. Prints 24 characters of the printer buffer contents (ASCII). After printing, the contents of the buffer remain unchanged.

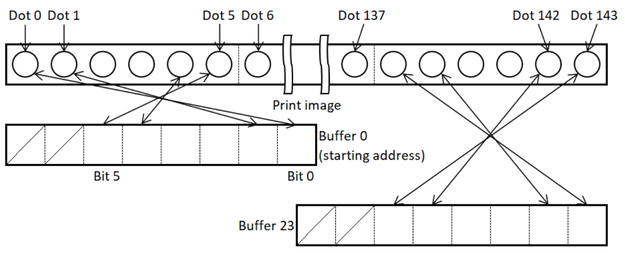
Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
		• Parameters – At entry ∗ `(X)`: Starting address of the buffer. Buffer size: 24 bytes. Data is in ASCII code. – At return ∗ `(C)`: abnormal I/O flag • Registers retained: `(A)`, `(B)`, `(X)`. • Subroutines referenced: – `SNSCOM`. – `NDFEED`. – `CHKSWT`. – `CHKRS`. • Variables used: `R0`, `R1`.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
PRTDOT	FF91	Prints one dot-line of bit-image data. One dot-line of bit-image print consists of 144 dots and is specified by the 24 bytes in the buffer. Data is entered into the buffer as follows.  Dot 0 Dot 1 Dot 5 Dot 6 Dot 137 Dot 142 Dot 143 Print image Buffer 0 (starting address) Bit 5 Bit 0 Buffer 23 Bits 6 and 7 of each byte of the buffer have no meaning. If during the printing of an image an empty interval occurs until this subroutine is called, a 1-dot blank line will result.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents

		• Parameters
		– At entry
		∗ `(X)`: Starting address of buffer.
		– At return
		∗ `(C)`: abnormal I/O flag
		• Registers retained: `(A)`, `(B)`, `(X)`.
		• Subroutines referenced:
		– `SNSCOM`.
		– `CHKSWT`.
		– `CHKRS`.
		• Variables used: `ROH`.
		• Example
		When the following is printed
		○ ● ● ○ ○ ○ ● ● ○ ○ ○ ○
		`LDX   BUFF`
		`JSR   PRTDOT`
		
		`BUFF  FCB    $06,$03,...`

`NDFEED`	`FF8F`	Performs paper feed for n dot-lines.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
		• Parameters – At entry ∗ `(A)`: Number of dot-lines of line feed performed. – At return ∗ `(C)`: abnormal I/O flag • Registers retained: `(A)`, `(B)`, `(X)`. • Subroutines referenced: – `SNSCOW`. – `CHKRS`. • Variables used: none.
`SCRCPY`	`FF8B`	Copies the data displayed on the LCD on the microprinter. The width of the LCD is 120 dots and that of the printer, 144 dots. The data is left-justified and the remaining 24 dots remain blank.

Continues in next page...

...continued from previous page.		
Subroutine name	Entry point	Contents
		• Parameters – At entry: none. – At return ∗ `(C)`: abnormal I/O flag • Registers retained: `(A)`, `(B)`, `(X)`. • Subroutines referenced: – `SNSCOW`. – `SNSCOM`. – `WRTP26`. – `CHKSWT`. – `LCDMOD`. • Variables used: none.

7.6 Microprinter work areas

Address (from)	(to)	Variable name	Byte count	Description
`190`	`195`	`CHRPTN`	6	Work area for character font (for 1 character).
`196`	`196`	`COLCNT`	1	Data count in buffer (0-24 bytes).
`197`	`1AE`	`CHRDAT`	24	Buffer data for 1 line of characters.

7.7 Sample listings: print full graphic pattern

```
   0 :                       ; PRINT
   0 :                       ; Print full grahic pattern
   0 :                       ; Print full graphic pattern to internal micro printer
   0 :                       ; By K.A.
   0 :                       ;
   0 :                              PAGE   0
   0 :                              CPU    6301
1000 :                              ORG    $1000
1000 :                       ;
1000 :                       ;
1000 : =$FF91                PRTDOT EQU    $FF91
1000 :                       ;
1000 :                       ; Print pattern of oblique lines.
1000 :                       ; .  .  .  .  .  .  .  .  .  .  .
1000 :                       ;  .  .  .  .  .  .  .  .  .  .  .
1000 :                       ;   .  .  .  .  .  .  .  .  .  .
1000 :                       ; .  .  .  .  .  .  .  .  .  .  .
1000 :                       ;  .  .  .  .  .  .  .  .  .  .  .
1000 :                       ;   .  .  .  .  .  .  .  .  .  .
1000 :                       ; .  .  .  .  .  .  .  .  .  .  .
1000 :                       ;
1000 :                       ;
1000 : 86 08                        LDAA   #8     ; (A): repeating times
1002 : C6 03                 PRTR10 LDAB   #3     ; (B): pattern number (3, 2, 1)
1004 :                       ;
1004 : CE 10 1F              PRTRPT LDX    #PATN1 ; Set address of print pattern
1007 : C1 03                        CMPB   #3     ; If (B)=3, pattern 1
1009 : 27 0A                        BEQ    PRTR30
100B : CE 10 37                     LDX    #PATN2 ; If (B)=2, pattern 2
100E : C1 02                        CMPB   #2
1010 : 27 03                        BEQ    PRTR30
1012 : CE 10 4F                     LDX    #PATN3 ; If (B)=1, pattern 3
```

```
1015 : BD FF 91            PRTR30 JSR    PRTDOT ; Print by graphic image
1018 : 5A                         DECB
1019 : 26 E9                      BNE    PRTRPT
101B : 4A                         DECA          ; Finished?
101C : 26 E4                      BNE    PRTR10
101E :                     ;
101E : 39                         RTS
101F :                     ;
101F : 09 09 09 09 09 09   PATN1  FCB    $09,$09,$09,$09,$09,$09
1025 : 09 09 09 09 09 09          FCB    $09,$09,$09,$09,$09,$09
102B : 09 09 09 09 09 09          FCB    $09,$09,$09,$09,$09,$09
1031 : 09 09 09 09 09 09          FCB    $09,$09,$09,$09,$09,$09
1037 : 12 12 12 12 12 12   PATN2  FCB    $12,$12,$12,$12,$12,$12
103D : 12 12 12 12 12 12          FCB    $12,$12,$12,$12,$12,$12
1043 : 12 12 12 12 12 12          FCB    $12,$12,$12,$12,$12,$12
1049 : 12 12 12 12 12 12          FCB    $12,$12,$12,$12,$12,$12
104F : 24 24 24 24 24 24   PATN3  FCB    $24,$24,$24,$24,$24,$24
1055 : 24 24 24 24 24 24          FCB    $24,$24,$24,$24,$24,$24
105B : 24 24 24 24 24 24          FCB    $24,$24,$24,$24,$24,$24
1061 : 24 24 24 24 24 24          FCB    $24,$24,$24,$24,$24,$24
1067 :                     ;
1067 :                     ;
1067 :                            END
```

Chapter 8

ROM cartridge

8.1 General

The ROM cartridge, which is provided as a plug-in option of the HX-20, can read 2K to 16K bytes of data from an external ROM memory via the I/O ports using its addressing counter and shift register. The addressing counter is incremental and its value can also be reset to `0`.

The ROM cartridge is designed for an output-only file as a ROM file to allow data output in this file format.

8.2 Configuration

Table 8.1 shows the I/O ports related to the ROM cartridge.

MCU	Port	Input/ Output	Description
Master MCU	`P17`	Input	ROM data (1 bit).
	`P266`	Output	Shift/load select (`0`: load; `1`: shift).
	`P267`	Output	Clock.
Slave MCU	`P20`	Input	ROM cartridge interface judgement.
	`P46`	Input	ROM cartridge interface judgement.
	`P42`	Output	Shift register clear [`0`: OFF (clear); `1`: ON (don't clear)].
Continues in next page...			

...continued from previous page

MCU	Port	Input/ Output	Description
	P43	Output	Power supply (0: OFF; 1: ON).
	P44	Output	Addressing counter clear. [0: OFF (clear); 1: ON (don't clear)].

Table 8.1: ROM cartridge I/O ports

The ROM cartridge is configured as shown in Figure 8.1. One byte of ROM data at the address indicated by the addressing counter is input to the shift register, which in turn transfers the ROM data to the master MCU.

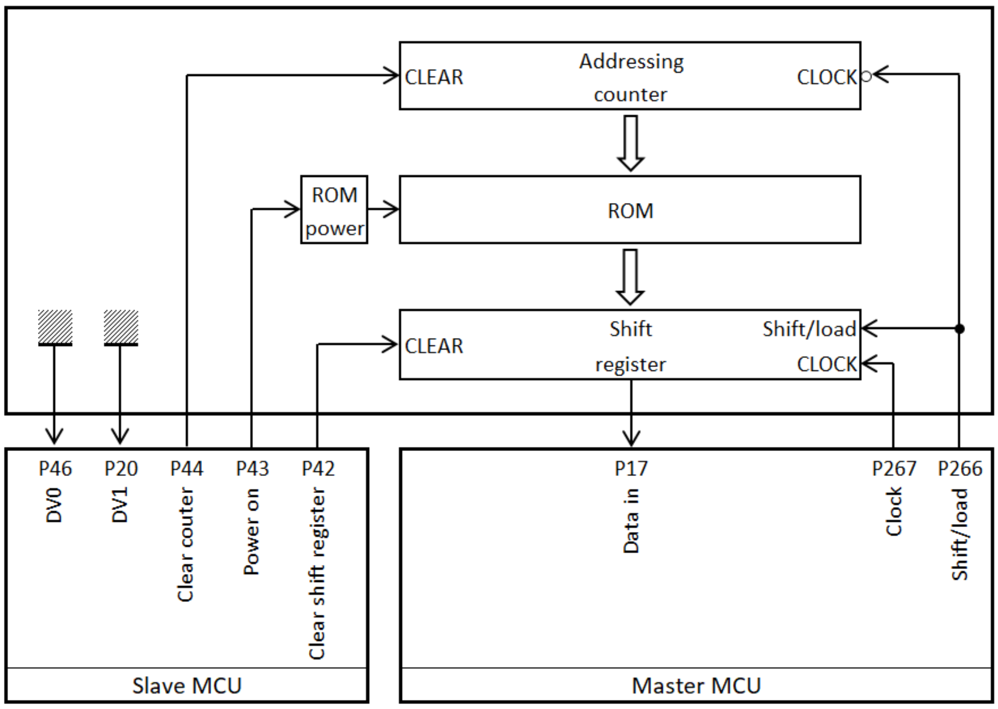


Figure 8.1: Block diagram of ROM cartridge

8.3 Data input procedure

Only two types of instructions are applicable to the addressing counter: Clear (by setting the `P44` of the slave MCU to '`0`') and Count-up. Data is fetched by the master MCU from the shift register by inputting one bit of data to the port `P17` of the master MCU each time the data bits in the shift register are moved. Data input from the ROM cartridge is performed by the procedure as detailed below.

1. The power supply of the ROM cartridge is turned ON.

   The port `P43` of the slave MCU is the power supply port to turn on or off the ROM cartridge. The master MCU instructs the slave MCU to issue a ROM Power ON command to turn on the power supply of the ROM cartridge.

2. The addressing counter is cleared.

   The addressing counter is automatically reset to `0` when the ROM Power ON command is issued to the ROM cartridge from the slave MCU.

3. The addressing counter is incremented to the address from which data is to be read.

   The counter counts up when the voltage level at the port `P266` (bit 6 at address `26`) of the master MCU changes from high to low.

4. When port `P266` is at low level, one byte of data at the address indicated by the addressing counter is loaded into the shift register at the leading edge of a `CLOCK` signal appearing at the `P267` (bit 7 at address `26`) of the master MCU. In this case, bit 7 is first loaded into the master MCU through port `P17` (Data in).

5. When port `P266` is at high level, the contents of the shift register are shifted one bit at the trailing edge of the `CLOCK` signal (`P267`). By repeating this operation 7 times, one byte of data can be fetched by the master MCU.

6. If data input from the ROM cartridge is no longer required, the power supply of the ROM cartridge must be turned off by sending a command from the master MCU to the slave MCU to turn off the ROM power supply.

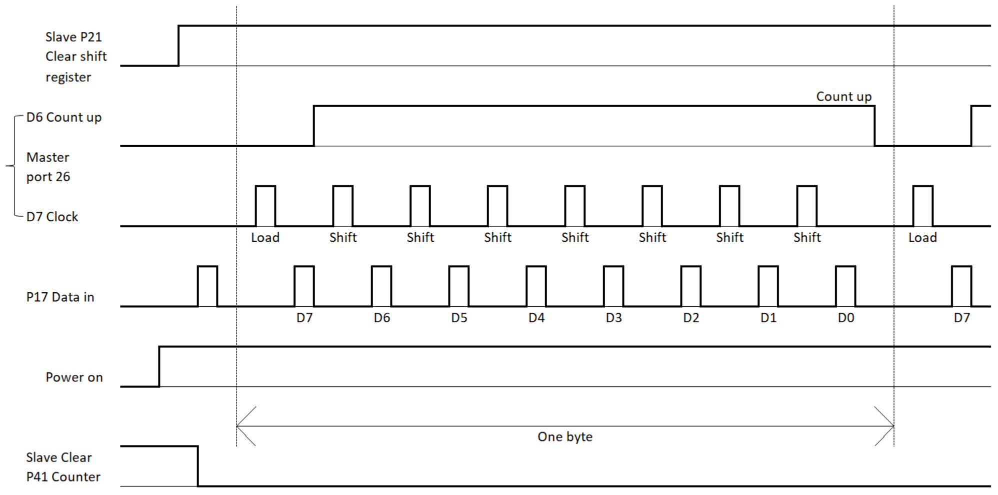


Figure 8.2: Timing chart of data input from ROM cartridge

Note: if data is input after clearing the shift register, the data that is input to the master MCU is binary `0`. If this Shift Registry Clear operation is performed when the optional microcassette drive is connected to the HX-20, binary `1` is input.

8.4 ROM file

Data input from the optional ROM cartridge is supported in the form of data input from a ROM file. The ROM file consists of 32 headers and a data area. Each header may contain a maximum of 32 bytes of data as header information. The ROM file may only be accessed sequentially but not randomly.

Figure 8.3 shows the structure of the ROM file.

Headers are allocated as fixed areas from address `0000` in units of 32 bytes. Header 0 is from address `0000` to address `001F`. A maximum of 32 headers can be set. The first one byte of each header represents the first letter of the filename as well as header information. If the first one byte of a header is "`00`", it indicates that the file with that header has been deleted. If "`FF`", if indicates that no subsequent header exists.

If the first one byte of header 2 is "`FF`", headers 0 and 1 are valid as headers. The contents of the header information are shown in Section 8.7 below.

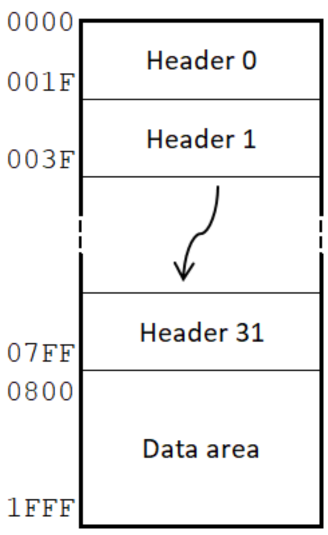


Figure 8.3: Structure of ROM file

8.5 Subroutines for ROM cartridge

The following 4 subroutines are provided for the ROM cartridge:

1. `OPNPRM`: opens the ROM file.
2. `REDPRM`: inputs data from the ROM file in units of one byte..
3. `CLSPRM`: closes the ROM file.
4. `DIRPRM`: inputs the ROM file directory.

8.6 File input procedure

A ROM file is processed for data input as follows:

1. Opening the ROM file

   Subroutine "`OPNPRM`" is used to start the input of data from the ROM file.

2. Data input

   Data is read from the ROM file in units of one byte by subroutine "`REDPRM`".

3. Closing the ROM file

   Data input from the ROM file is terminated by subroutine "`CLSPRM`".

Note: upon opening the ROM file, the ROM cartridge is energized. The ROM file must be closed soon after the data input has been completed particularly when an NMOS type PROM with high power consumption is used.

8.7 Header format of ROM file

Columns		Bytes	Item	Description
from	to			
0	7	8	Filename	Filename (in ASCII codes). Column 0 represents `ID` in addition to the filename • `00`: file has been deleted. • `FF`: no subsequent header exists.
8	15	8	File type	File type (in ASCII codes).
16	19	4	Starting address	The starting address of the ROM area secured as a file. The binary address value is expressed in 4-digit hexadecimal numbers (ASCII codes).
20	23	4	Ending address+1	The address next to the ending address of the ROM area secured as a file. The binary address value is expressed in 4-digit hexadecimal numbers (ASCII codes).
24	29	6	Date	Month, day and year each expressed in 2-digit ASCII codes.
30	31	2		Unused.

8.8 ROM cartridge subroutine table

Subroutine name	Entry point	Description
`OPNPRM`	`FEEC`	ROM file input open.
		• Parameters – At entry ∗ `(A)`: this parameter specifies whether or not the filename is to be returned. · `01`: return the filename opened in the packet. · `00`: do not return the filename opened in the packet. ∗ `(X)`: starting address of packet. – Packet 1. Filename (8 bytes). 2. File type (8 bytes). 3. Filename (8 bytes): enter the filename opened when the filename is to be returned. 4. File type (8 bytes): enter the filen type opened when the filename is to be returned. **Note:** in the filename specification for the packet, if the string specifying a filename contains an asterisk (`*`), the filename matching terminates at the point of the asterisk and the system assumes that both the filenames have matched. In BASIC version 1.0, when the matching of the filename with an asterisk (`*`) terminates, the system assumes that both the file types have also matched (note that the ROM file open procedure differs from the cassette file open procedure).

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
		• Parameters *(continued)* – At return ∗ `(C)`: abnormal I/O flag. ∗ `(A)`: return codes. · `00`: normal. · `A0`: ROM cartridge not connected. · `A3`: file not found. · `A2`: file already open. · `A4`: invalid data header format. · `A5`: invalid header address format. ∗ `(Z)`: according to the value of `(A)`. • Registers retained: none. • Subroutines referenced: – `PRMPON`. – `PREDBY`. – `HEXBIN`. – `CLSPRM`. • Variables used: `R0`, `R1` and `R2`.
`REDPRM`	`FEE9`	Input of one byte from ROM file.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
		• Parameters – At entry: none. – At return ∗ `(C)`: abnormal I/O flag. ∗ `(A)`: input data. ∗ `(B)`: return codes. · `00`: normal. · `01`: end of file. · `A3`: file not opened. ∗ `(Z)`: according to the value of `(B)`. • Registers retained: `(X)`. • Subroutines referenced: – `ADSTEP`. • Variables used: none.
`CLSPRM`	`FEE6`	ROM file close.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents

		• Parameters
		  – At entry: none.
		  – At return
		    ∗ `(C)`: abnormal I/O flag.
		• Registers retained: `(B)` and `(X)`.
		• Subroutines referenced:
		  – `CHKRS`.
		  – `SNSCOM`.
		Note: an attempt to close an unopened ROM file is not regarded by the system as an error.

`DIRPRM`	`FEE3`	ROM file directory read. This subroutine specifies record number of the directory and inputs the record.

Continues in next page...

...*continued from previous page.*

Subroutine name	Entry point	Contents
		• Parameters – At entry ∗ `(A)`: directory record number from 0 through 63 (decimal). ∗ `(X)`: starting address of memory locations where the directory record is stored. The size of each record must be 32 bytes. – At return ∗ `(C)`: abnormal I/O flag. ∗ `(A)`: return codes · `00`: normal. · `A0`: ROM cartridge not connected. · `A3`: invalid specification of the directory record number. ∗ `(Z)`: according to the value of `(A)`. • Registers retained: none. • Subroutines referenced: – `PRMPON`. – `ADSTEP`. – `PREDBY`. – `CLSPRM`.

8.9 ROM cartridge work areas

Address (from)	Address (to)	Variable name	Byte count	Description
208	208	PRMSTS	1	Status of the ROM file • Bit 0: file open status flag – 0: file not opened. – 1: file opened. • Bits 1 to 6: undefined. • Bit 7: Power supply for ROM – 0: off. – 1: on.
209	20A	STAPRS	2	ROM addressing counter.
20B	20C	FTADRS	2	Starting address of file.
20D	20E	EDADRS	2	Ending address of file+1.

8.10 Sample listings: ROM cartridge interface routine

```
0 :                  ; ROMOPT
0 :                  ; ROM cartridge interface routine
0 :                  ; By K.A.
0 :                  ;
0 :                         PAGE   0
0 :                         CPU    6301
0 :                  ;
0 :                  ; Common definition
0 :                  ;
0 :                  ; MPU 6301 I/O port
0 : =$2              PORT1  EQU    $02    ; I/O port 1 (address)
0 : =$3              PORT2  EQU    $03    ; I/O port 2 (address)
0 :                  ;
0 :                  ; Other registers
0 :                  ; Register meanings
0 :                  ; PORT1 $02
0 :                  ;      0:R Data set ready (0:high 1:low)
0 :                  ;      1:R Clear to send (0:high 1:low)
0 :                  ;      2:R Port to slave P34 (SFLAG)
0 :                  ;      3:R Interrupt from external port (0:interrupt)
0 :                  ;      4:R Power fail (0:abnormal)
0 :                  ;      5:R Keyboard interrupt flag (0:interrupt)
0 :                  ;      6:R Peripheral status (0:high 1:low) (from serial)
0 :                  ;      7:R Micro cassette counter / micro cassette exists
0 :                  ;
0 :                  ;      $26
0 :                  ;      0:W LCD command/data 1
0 :                  ;      1:W LCD command/data 2
0 :                  ;      2:W LCD command/data 4
0 :                  ;      3:W LCD command/data selection (0:data 1:command)
0 :                  ;      4:W Keyboard interrupt mask (0:close 1:open)
0 :                  ;      5:W Peripheral control (to serial)
```

```
  0 :                   ;      6:W To plug in 1
  0 :                   ;      7:W To plug in 2 and slave P40
  0 :                   ;
  0 :                   ; Common definition
  0 :                   ;
  0 :                   ; Zero page RAM
 4E :                          ORG    $4E
 4E :                   PWRFLG RMB    1      ; Bit 0-3: clock power on mode
 4F :                   ;                    ;      $01: power on by clock in BASIC mode.
 4F :                   ;                    ;      $02: power on by clock in application mode.
 4F :                   ;                    ; Bit 4-7: before power off, call procedure mode
 4F :                   ;                    ;      $01: before power off, call procedure in
 4F :                   ;                    ;           BASIC mode.
 4F :                   ;                    ;      $02: before power off, call procedure in
 4F :                   ;                    ;           application mode.
 4F :                   ;
 4F :                   P26    RMB    1      ; Value of address $26
 50 :                   ; General registers used by I/O routine
 50 : =$50              R0     EQU    *      ; 2 bytes register (R0H,R0L)
 50 :                   R0H    RMB    1
 51 :                   R0L    RMB    1
 52 : =$52              R1     EQU    *      ; 2 bytes register (R1H,R1L)
 52 :                   R1H    RMB    1
 53 :                   R1L    RMB    1
 54 : =$54              R2     EQU    *      ; 2 bytes register (R2H,R2L)
 54 :                   R2H    RMB    1
 55 :                   R2L    RMB    1
 56 : =$56              R3     EQU    *      ; 2 bytes register (R3H,R3L)
 56 :                   R3H    RMB    1
 57 :                   R3L    RMB    1
 58 :                   ;
 7C :                          ORG    $7C
```

```
  7C :                   SIOSTS RMB    1      ; Slave I/O status (each bit 0:off, 1:on)
  7D :                   ;                    ; Bit 0: printer
  7D :                   ;                    ; Bit 1: external cassette
  7D :                   ;                    ; Bit 2: internal cassette
  7D :                   ;                    ; Bit 3: RS232C on (read)
  7D :                   ;                    ; Bit 4: speaker on
  7D :                   ;                    ; Bit 5: PROM cassette
  7D :                   ;                    ; Bit 6: bar code reader
  7D :                   ;                    ; Bit 7: break slave CPU (0: on execute;
  7D :                   ;                                            1: broken by interrupt)
  7D :                   MIOSTS RMB    1      ; Main I/O status (each bit 0:off, 1:on)
  7E :                   ;                    ; Bit 0: LCD on read/write characters
  7E :                   ;                    ; Bit 1: now sending command to slave CPU
  7E :                   ;                    ; Bit 2: now transmitting data to serial (1:on)
  7E :                   ;                    ; Bit 3: on clock interrupt (1:on)
  7E :                   ;                    ; Bit 4: (power fail)
  7E :                   ;                    ; Bit 5: (off power switch)
  7E :                   ;                    ; Bit 6: on pause key
  7E :                   ;                    ; Bit 7: on break key
  7E :                   ; ROM cassette work area
 208 :                          ORG    $208
 208 : =$208             PRWKTP EQU    *      ; ROM work top
 208 :                   PRMSTS RMB    1      ; ROM status (bit 7: power on 1:on, o:off
 209 :                   ;                    ;             bit 0: open flag 1:open, 0:close)
 209 :                   STADRS RMB    2      ; ROM address counter
 20B :                   FTADRS RMB    2      ; Address of top of file
 20D :                   EDADRS RMB    2      ; Address of last of file + 1
 20F :                   ;
1000 :                          ORG    $1000
1000 :                   ;
1000 : =$FF2E            CHKPLG EQU    $FF2E
1000 : =$FF19            SNSCOM EQU    $FF19
```

```
 97/    1000 : =$FF16              CHKRS  EQU    $FF16
 98/    1000 : =$FED4              WRTP26 EQU    $FED4
 99/    1000 : =$FF2B              HEXBIN EQU    $FF2B
100/    1000 :                     ;
101/    1000 : =$51                CMPRON EQU    $51    ; ROM power on command to slave MCU
102/    1000 : =$52                CMPROF EQU    $52    ; ROM power off command to slave MCU
103/    1000 :                     ;
104/    1000 : =$12E               FILBYT EQU    $12E   ; Rest bytes in the file (2 bytes size)
105/    1000 :                     ;
106/    1000 :                     ;
107/    1000 :                     ; Header format of PROM
108/    1000 :                     ;      00 - 07 (dec) : file name (00: $00:deleted; $FF:end of header)
109/    1000 :                     ;      08 - 15       : file type
110/    1000 :                     ;      16 - 19       : top address of the file
111/    1000 :                     ;      20 - 23       : bottom address + 1
112/    1000 :                     ;      24 - 29       : date
113/    1000 :                     ;      30 - 31       : not used
114/    1000 :                     ;
115/    1000 :                     ; Function: open to read
116/    1000 :                     ; On entry
117/    1000 :                     ;      (A)=read mode (0:not answer file name
118/    1000 :                     ;                     1:answer file name)
119/    1000 :                     ;      (X)=packet address
120/    1000 :                     ;             Packet 0-7 : file name
121/    1000 :                     ;                    8-15: file type
122/    1000 :                     ; On exit
123/    1000 :                     ;      (A)=return code
124/    1000 :                     ;             $00:normal
125/    1000 :                     ;             $A0:without ROM cassette
126/    1000 :                     ;             $A1:file is not found
127/    1000 :                     ;             $A2:already open
128/    1000 :                     ;             $A3:directory number error
```

```
1000 :              ;             $A4:ROM format error
1000 :              ;             $A5:addressing error
1000 :              ;       (C)=0
1000 :              ;       (Z)=depends on value of (A)
1000 :              ;       Packet
1000 :              ;             16-23:found file name (when "answer file name" mode)
1000 :              ;             24-31:found file type
1000 :              ; Register preserve
1000 :              ;       None
1000 :              ;
1000 :              ; Work area as register
1000 :              ;       R0 : save packet address
1000 :              ;       R1H: save mode when open procedure was called (value of (A))
1000 :              ;       R1L: the flag whether found file name is matched
1000 :              ;             Bit 7: stop to compare (0:continue to compare, 1:stop)
1000 :              ;             Bit 0-4: flag file name is matched (0:matched, others:no)
1000 :              ;       R2H: read character (read byte routine)
1000 :              ;       R2L: header number
1000 :              ;
1000 : 97 52        OPNPRM STAA   R1H           ; Save mode "answer file name or not"
1002 : DF 50               STX    R0            ; Save packet address
1004 :              ;
1004 : BD 10 EC            JSR    PRMPON        ; With ROM cartridge? (reset address counter)
1007 : 26 7B               BNE    OPNP67        ; If nonzero, error detect
1009 : 97 55               STAA   R2L           ; Header number = 0
100B : 86 81               LDAA   #$81          ; Set open and power on flag
100D : B7 02 08            STAA   PRMSTS
1010 :              ;
1010 :              ; Read header and search file name
1010 : 5F           OPNP20 CLRB                 ; (B): data counter (0 - $0F)
1011 : D7 53               STAB   R1L           ; Flag (name is matched)
1013 : DE 50        OPNP25 LDX    R0            ; (X): packet address
```

```
1015 : 3A                       ABX
1016 : BD 10 C2                 JSR    PREDBY        ; Read one character from the ROM
1019 : 25 72                    BCS    OPNP80
101B : 5D                       TSTB                 ; Address = first column of file name?
101C : 26 07                    BNE    OPNP26
101E : 81 FF                    CMPA   #$FF          ; Not found? (last directory mark = $FF)
1020 : 27 6C                    BEQ    OPNP90
1022 : 4D                       TSTA                 ; Deleted?    (deleted file mark = $00)
1023 : 27 2A                    BEQ    OPNP35
1025 : 7D 00 52          OPNP26 TST    R1H           ; "Answer file name" mode?
1028 : 27 02                    BEQ    OPNP27
102A : A7 10                    STAA   16,X          ; Yes, store file name to data packet.
102C : 7D 00 53          OPNP27 TST    R1L           ; Stop to compare (file name is matched)?
102F : 2B 14                    BMI    OPNP29
1031 : 36                       PSHA                 ; '*': mark to stop to compare
1032 : 86 2A                    LDAA   #'*'
1034 : A1 00                    CMPA   0,X
1036 : 32                       PULA
1037 : 26 05                    BNE    OPNP28
1039 : 72 80 53                 OIM    #$80,R1L      ; '*': mark. Set "stop compare" bit
103C : 20 07                    BRA    OPNP29
103E :                   ;
103E : A1 00             OPNP28 CMPA   0,X           ; Compare file name
1040 : 27 03                    BEQ    OPNP29
1042 : 7C 00 53                 INC    R1L           ; Set "file not matched" flag
1045 :                   ;
1045 : 5C                OPNP29 INCB                 ; Finish to compare?
1046 : C1 10                    CMPB   #16           ; File name and file type are 16 bytes long
1048 : 26 C9                    BNE    OPNP25
104A :                   ; File name and file type are completed to compare
104A : 7B 0F 53                 TIM    #$F,R1L       ; OK?
104D : 27 12                    BEQ    OPNP50
```

```
104F :                    ; No, compare next header
104F : 7C 00 55           OPNP35 INC    R2L           ; R2L: header number (next)
1052 : D6 55                     LDAB   R2L           ; Address of header = '32' * 'header number'
1054 : C1 40                     CMPB   #64
1056 : 2A 36                     BPL    OPNP90        ; Limit of the header ($000 - $3FF)
1058 : 86 20                     LDAA   #32
105A : 3D                        MUL
105B : 18                        XGDX                 ; (X): next addressing pointer
105C :                    ;
105C : BD 11 55                  JSR    ADSTEP        ; Set addressing counter to first column
105F : 20 AF                     BRA    OPNP20        ;   of the header
1061 :                    ;
1061 :                    ; Top address and last address which are shown by ASCII code are
1061 :                    ; converted to binary value
1061 : CE 02 04           OPNP50 LDX    #PRWKTP-4
1064 : 8D 5C              OPNP65 BSR    PREDBY        ; (A,B) <- ASCII coded hexadecimal value
1066 : 36                        PSHA
1067 : 8D 59                     BSR    PREDBY
1069 : 16                        TAB
106A : 32                        PULA
106B : BD FF 2B                  JSR    HEXBIN        ; Convert hex to binary
106E : 26 15                     BNE    OPNP70        ; Error?
1070 : A7 07                     STAA   FTADRS-PRWKTP+4,X
1072 : 08                        INX
1073 : 8C 02 08                  CPX    #PRWKTP
1076 : 26 EC                     BNE    OPNP65
1078 :                    ;
1078 : EC 05                     LDD    EDADRS-PRWKTP,X ; 'EDADRS' <- Last address
107A : A3 03                     SUBD   FTADRS-PRWKTP,X ; 'FTADRS' <- Top address
107C : FD 01 2E                  STD    FILBYT        ; 'FILBYT' <- Data number in the file
107F :                    ;
107F : 86 81                     LDAA   #$81          ; Set "opened file" flag
```

```
1081 : A7 00                  STAA   PRMSTS-PRWKTP,X
1083 : 4F                     CLRA
1084 : 39              OPNP67 RTS
1085 :                 ;
1085 : 86 A4           OPNP70 LDAA   #$A4          ; Format error
1087 : 36              OPNP75 PSHA
1088 : BD 11 3D               JSR    CLSPRM        ; Error close
108B : 32                     PULA
108C : 16                     TAB                  ; Set (Z), (N)
108D : 39              OPNP80 RTS
108E :                 ;
108E : 86 A1           OPNP90 LDAA   #$A1          ; Return code (file was not found)
1090 : 20 F5                  BRA    OPNP75
1092 :                 ;
1092 :                 ;
1092 :                 ; Function: read one character from file
1092 :                 ; On entry
1092 :                 ;      None parameter
1092 :                 ; On exit
1092 :                 ;      (A)=read data
1092 :                 ;      (B)=status
1092 :                 ;             $00: normal
1092 :                 ;             $01: end of file
1092 :                 ;             $A3: file not open
1092 :                 ;      (C)=0
1092 :                 ;      (Z)=depends on value of (B)
1092 :                 ; Register preserve
1092 :                 ;      (X)
1092 :                 ;
1092 : 3C              REDPRM PSHX                 ; Save (X)
1093 :                 ;
1093 : C6 A3                  LDAB   #$A3          ; Preset error code (file not open)
```

```
1095 : B6 02 08                 LDAA   PRMSTS        ; Is power on? (bits 0, 7 both on)
1098 : 2A 0D                    BPL    REDP08
109A : 47                       ASRA
109B : 24 0A                    BCC    REDP08
109D :                   ;
109D : FC 02 0D          REDP05 LDD    EDADRS        ; Is current address bottom in the file?
10A0 : B3 02 0B                 SUBD   FTADRS
10A3 : 26 06                    BNE    REDP10
10A5 :                   ;
10A5 : C6 01                    LDAB   #1            ; EOF return
10A7 : 4F                REDP08 CLRA
10A8 : 5D                       TSTB                 ; Set (Z), (N), clear (C)
10A9 : 38                       PULX
10AA : 39                       RTS
10AB :                   ; Read on bytes from file
10AB : =$10AB            REDP10 EQU    *
10AB : 18                       XGDX
10AC : 09                       DEX
10AD : FF 01 2E                 STX    FILBYT        ; Set "reset data number in the file"
10B0 : C6 A5                    LDAB   #$A5          ; Preset "addressing error" flag
10B2 : 25 F3                    BCS    REDP08
10B4 : FE 02 0B                 LDX    FTADRS        ; Non error
10B7 : 3C                       PSHX
10B8 : BD 11 55                 JSR    ADSTEP        ; ROM addressing <- +1 increment
10BB : 38                       PULX
10BC : 08                       INX
10BD : FF 02 0B                 STX    FTADRS        ; Addressing counter <- +1 increment
10C0 : 5F                       CLRB                 ; Return code = normal
10C1 : 38                       PULX
10C2 :                   ;
10C2 :                   ;
10C2 :                   ; Entry point "read next one byte"
```

```
10C2 :                     ; On entry
10C2 :                     ;      Parameter: none
10C2 :                     ;      Read one byte and increment addressing counter
10C2 :                     ; On exit
10C2 :                     ;      (A): read character
10C2 :                     ; Register preserve
10C2 :                     ;      (B), (X)
10C2 :                     ; Work as register
10C2 :                     ;      R2H: counter for 8 times and work area for read data
10C2 :                     ;      R2H    C      Bit7                  Bit0
10C2 :                     ;                     0  0  0  0  0  0  0  1
10C2 :                     ;                            |
10C2 :                     ;                            V
10C2 :                     ;             0       0  0  0  0  0  0  1  X
10C2 :                     ;                            |            X:read bit
10C2 :                     ;                            V
10C2 :                     ;             0       0  0  0  0  0  1  X  X
10C2 :                     ;                                         X:read bit
10C2 :                     ;
10C2 : 37                  PREDBY PSHB
10C3 : 86 01                      LDAA   #$1           ; Mark for 8th times
10C5 : 97 54                      STAA   R2H
10C7 : 5F                         CLRB
10C8 : C4 7F               REDP20 ANDB   #$FF-$80      ; Bit 7 low (D7)
10CA : 86 C0                      LDAA   #$C0          ; Bit 6, 7 effective
10CC : BD FE D4                   JSR    WRTP26        ; Clock low (first time: D6 low)
10CF : CA 80                      ORAB   #$80          ; Clock high (first time: read data
10D1 : BD FE D4                   JSR    WRTP26        ;              second time: shift data)
10D4 :                     ;
10D4 : 96 02                      LDAA   PORT1         ; Input data (bit7, bit6,...)
10D6 : 48                         ASLA
10D7 : 79 00 54                   ROL    R2H           ; R2L: shift one bit which was get
```

```
10DA : CA 40                  ORAB   #$40          ; For D6: high
10DC : 24 EA                  BCC    REDP20        ; Complete to read 8 bits?
10DE :                 ;
10DE : FC 02 09               LDD    STADRS        ; Addressing pointer <- +1 increment
10E1 : C3 00 01               ADDD   #1
10E4 : FD 02 09               STD    STADRS
10E7 :                 ;
10E7 : 96 54                  LDAA   R2H           ; (A) <- read data
10E9 : 33                     PULB
10EA : 5D                     TSTB                 ; Clear (C), set (Z) for "REDPRM" routine
10EB : 39                     RTS
10EC :                 ;
10EC :                 ; Power on ROM
10EC :                 ; Procedure
10EC :                 ;      1: Check plugin option (ROM)?
10EC :                 ;      2: Clear addressing counter
10EC :                 ;      3: Power on
10EC :                 ; Parameter
10EC :                 ;      On entry: none
10EC :                 ;      On exit
10EC :                 ;             (A): return code (00:normal, others:error)
10EC :                 ;             (C): I/O error flag
10EC :                 ;             (Z): depends on value of (A)
10EC :                 ; Register preserve
10EC :                 ;      (X)
10EC :                 ;
10EC : BD FF 2E        PRMPON JSR    CHKPLG        ; Check plug-in option
10EF : 25 26                  BCS    PRMP80
10F1 : 16                     TAB
10F2 : 26 23                  BNE    PRMP80
10F4 : 72 20 7C               OIM    #$20,SIOSTS   ; Slave ROM cassette on
10F7 : FD 02 09               STD    STADRS        ; ROM address = 0 (A,B)=0
```

```
10FA : 86 C0                 LDAA   #$C0
10FC : BD FE D4              JSR    WRTP26        ; Set D6, D7 low (count, clock)
10FF : 86 51                 LDAA   #$51
1101 : BD FF 19              JSR    SNSCOM        ; Send "PROM on command" to slave MCU
1104 : 25 11                 BCS    PRMP80
1106 : 3C                    PSHX
1107 : CE 01 90              LDX    #400         ; Wait 2ms
110A : 09             PRMP20 DEX
110B : 26 FD                 BNE    PRMP20
110D : FE 02 08              LDX    PRMSTS        ; Set power on flag (on bit7)
1110 : 62 80 00              OIM    #$80,0,X
1113 : 38                    PULX
1114 : 4F                    CLRA
1115 : 20 33                 BRA    CLSP10        ; (JMP CHKRS)
1117 :                ;
1117 : 86 A0          PRMP80 LDAA   #$A0          ; Without ROM cassette (error)
1119 : 39                    RTS
111A :                ;
111A :                ;
111A :                ; Function: read directory
111A :                ; On entry
111A :                ;      (A): directory number (0 to 63)
111A :                ;      (X): address where header are stored
111A :                ; On exit
111A :                ;      (A): return code
111A :                ;             $00: normal
111A :                ;             $A0: without ROM cassette
111A :                ;             $A3: directory number error
111A :                ;      (C): 0
111A :                ;      (Z): depends on value of (A)
111A :                ; Register preserve: none
111A :                ;
```

```
111A : 16              DIRPRM TAB                  ; Save directory number
111B : 86 A3                  LDAA   #$A3          ; (A) <- Directory error flag (preset)
111D : C1 40                  CMPB   #64           ; Is directory number limit (00 - 63) OK?
111F : 24 29                  BCC    CLSP10
1121 :                 ;
1121 : DF 50                  STX    R0            ; Save address of directory
1123 :                 ;
1123 : 37                     PSHB
1124 : 8D C6                  BSR    PRMPON        ; Power on (check PROM)
1126 : 33                     PULB
1127 : 26 21                  BNE    CLSP10
1129 :                 ;
1129 : 86 20                  LDAA   #32           ; Calculate header address (32 * 'number')
112B : 3D                     MUL
112C : 18                     XGDX
112D : 8D 26                  BSR    ADSTEP        ; Set ROM address
112F : C6 20                  LDAB   #32
1131 : DE 50                  LDX    R0
1133 : 37              DIRP10 PSHB
1134 : 8D 8C                  BSR    PREDBY        ; Read one character
1136 : A7 00                  STAA   0,X
1138 : 08                     INX
1139 : 33                     PULB
113A : 5A                     DECB
113B : 26 F6                  BNE    DIRP10
113D :                 ;
113D :                 ;
113D :                 ; Function: close ROM cassette
113D :                 ; On entry
113D :                 ;      Parameter none
113D :                 ; On exit
113D :                 ;      (C): I/O error flag
```

```
113D :                  ; Register preserve
113D :                  ;      (B), (X)
113D :                  ;
113D : 7F 02 08         CLSPRM CLR    PRMSTS        ; Set ROM status "power off", "closed file"
1140 : 86 52                   LDAA   #CMPROF       ; Send "power off command" to slave CPU
1142 : BD FF 19                JSR    SNSCOM
1145 : 71 DF 7C                AIM    #$FF-$20,SIOSTS; Set flag ("ROM cassette is off")
1148 : 86 00                   LDAA   #0            ; (do not change (C) bit)
114A : 7E FF 16         CLSP10 JMP    CHKRS         ; Recover RS232 (open to read RS232)
114D :                  ;
114D :                  ;
114D :                  ; Function: set PROM address to destinated value
114D :                  ; On entry
114D :                  ;      (X): target address
114D :                  ; On exit
114D :                  ;      (C): I/O error flag
114D :                  ; Register preserve
114D :                  ;      None
114D :                  ;
114D : =$114D           ADST00 EQU    *
114D : 8D 9D                   BSR    PRMPON        ; Without ROM? (clear addressing counter)
114F : 26 20                   BNE    ADST80        ; Without?
1151 : 5F                      CLRB                 ; If ROM (A):0
1152 : FD 02 09                STD    STADRS        ; ROM addressing counter <- 0
1155 :                  ; Entry point of "ADSTEP" routine
1155 : 3C               ADSTEP PSHX                 ; (A,B) <- (X)
1156 : 32                      PULA
1157 : 33                      PULB
1158 : B3 02 09                SUBD   STADRS        ; New address >= current address?
115B : 27 14                   BEQ    ADST80        ; = ?
115D : 25 EE                   BCS    ADST00
115F :                  ;                           ; Case of "target address > current address"
```

```
115F : FF 02 09                 STX    STADRS        ; Set new address to "STADRS"
1162 : 18                       XGDX                 ; (X) <- Step count
1163 :                   ;
1163 : 5F                ADST30 CLRB                 ; Count up addressing counter
1164 : 86 C0                    LDAA   #$C0
1166 : BD FE D4                 JSR    WRTP26
1169 : C6 40                    LDAB   #$40
116B : BD FE D4                 JSR    WRTP26
116E : 09                       DEX
116F : 26 F2                    BNE    ADST30
1171 : 39                ADST80 RTS
1172 :                   ;
1172 :                          END
```

Chapter 9

Load module

9.1 General

The module format for output of data by the `SAVEM` command in BASIC or the `W` command in the Monitor is a special format calles a “Binary Load Module format”. One file is diveided into a number of records each containing memory addresses and data (Figure 9.1).

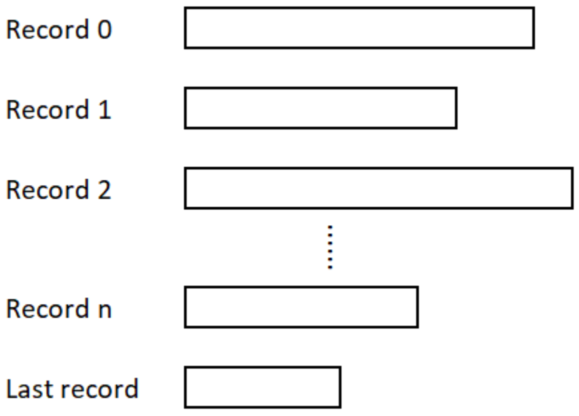


Figure 9.1: Division of file into records

Each record has a maximum length of 259 bytes and each data contained in the record is represented in binary numbers in units of one byte. The format of each record is shown below.

9.2 Load module (machine language) format

9.2.1 Intermediate record

Column	Size (bytes)	Item	Description
0	1	Record length	Indicates the length of the data contained in the record in binary numbers (`00` through `FF`).
1-2	2	Address	Indicates the address of the first data in the record in binary numbers `0000` through `FFFF` (in order of the upper and lower digits).
3	1	Data	Data 1. Namely, first data (`00` through `FF`).
4	1	Data	Data 2.
...			
$n+2$	1	Data	Data n (n must be a value in the range 0 to 255).
$n+3$	1	Checksum	This value must be such that the low-order 8 bits of the sum of the data values in columns 0 through $n+3$ becomes 0.

9.2.2 Last record

Column	Size (bytes)	Item	Description
0	1	Record length	This value must always be `0`.
1-2	2	Address	Indicates the entry point of a program in binary numbers (`0000` through `FFFF` in order of the upper and lower digits).
3	1	Checksum	This value must be such that the low-order 8 bits of the sum of the data values in columns 0 through 3 becomes 0.

9.3 Dump/load procedures

9.3.1 I/O devices

The basic I/O routines support the following devices:

1. Input
    (a) External audio cassette.
    (b) Built-in microcassette.
    (c) ROM cartridge.
2. Output
    (a) External audio cassette.
    (b) Built-in microcassette.

9.3.2 Dump/load procedures

The memory contents in the binary load module format are transferred to and from an external storage as follows:

1. Output to the external storeage
    (a) File opening
    Subroutine "`OPNDMP`" is provided to open the specified file (device) for output. Subroutine "`OPNWCS`" is called if the specified file is an external audio cassette.
    (b) Output of the memory contents
    Subroutine "`BIDUMP`" is provided to output the memory contents in the binary load module format to the opened file and closes it upon completion of the dumping.
2. Input from the external storage
    (a) File opening
    Subroutine "`OPNLOD`" is provided to open the specified file (device) for input. Subroutine "`OPNPRM`" is called if the specified file is a ROM cartridge.

(b) Loading into memory

Subroutine "BILOAD" is provided to store the input data in the binary load module format in the main memory and closes the file upon completion of the loading.

9.4 Binary dump/load subroutine table

Subroutine name	Entry point	Description
OPNDMP	FEE0	Binary memory dump open. This subroutine opens the file to be dumped in a binary absolute format and supports an external cassette and the built-in microcassette drive.

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
		• Parameters – At entry ∗ `(X)`: top address of a data packet. ∗ `(B)`: device name · ‘`M`’: microcassette drive. · ‘`C`’: external audio cassette. Packet 1. Interblock tape stop mode (1 byte) for external audio cassette or microcassette ∗ `00`: stop the tape between blocks. ∗ `01`: do not stop the tape between blocks. 2. Top address of buffer (2 bytes). The buffer size is 260 bytes. 3. Filename (8 bytes). 4. File type (8 bytes). 5. Dump start address (2 bytes). 6. Dump end address (2 bytes). 7. Offset value (2 bytes). 8. Program entry point (2 bytes). **Note:** the offset value is added to the dump start address, dump end address, or the program entry point as an unsigned binary number. – At return ∗ `(C)`: abnormal I/O flag. ∗ `(A)`: return code (this parameter is dependent on subroutines `OPNWCS` and `OPNWMS`.

Continues in next page...

...continued from previous page.		
Subroutine name	Entry point	Contents
		• Registers retained: none. • Subroutines referenced: – `OPNWMS`. – `OPNWCS`. • Variables used: `R0`, `R1`, `R2`, `R3`, `R4`, `R5`, `R6` and `R7`.
`BIDUMP`	`FEDD`	Binary memory dump. This subroutine dumps the memory contents in a binary absolute format to the file opened by subroutine `OPNDMP` and closes the file upon completion of the dumping.
		• Parameters – At entry: none – At return: depends on subroutines `WRTCS`, `WRTMS`. • Registers retained: none. • Subroutines referenced: – `WRTMS`. – `WRTCS`. • Variables used: `R0`, `R1`, `R2`, `R3`, `R4`, `R5`, `R6` and `R7`.

9.5 Binary dump/load work area

Address (from)	(to)	Variable name	Byte count	Description
20F	210	DLTPAD	2	First dump address.
211	212	DLBTAD	2	Last dump address.
213	214	DLOFAD	2	Offset value.
215	216	DLSTAD	2	Program entry point.
217	217	DLDVID	1	Dump/load device.
218	218	DLSTS	1	Status work area (dummy).
219	21A	DLDVIX	2	Table address of a dump/load routine.

9.6 Sample listings

```
 1/       0 :                     ; CLOCK
 2/       0 :                     ; Display current time on the physical screen
 3/       0 :                     ; MPU is sleep if clock update is not caused.
 4/       0 :                     ;
 5/       0 :                     ; By K.A.
 6/       0 :                     ;
 7/       0 :                            PAGE   0
 8/       0 :                            CPU    6301
 9/       0 :                     ;
10/       0 :                     ; Subroutine entry point
11/       0 : =$FFA9              SLEEP  EQU    $FFA9  ; Sleep CPU
12/       0 : =$FF4C              DSPLCH EQU    $FF4C  ; Display one character on the physical screen
13/       0 : =$FF49              DSPLCN EQU    $FF49  ; Display some characters on the physical screen
14/       0 :                     ;
15/       0 :                     ;
16/    1000 :                            ORG    $1000
17/    1000 :                     ;
18/    1000 : C6 00                      LDAB   #0            ; Clear screen
19/    1002 : BD FF 49                   JSR    DSPLCN
20/    1005 : 86 FF                      LDAA   #$FF          ; Alarm interrupt time
21/    1007 : 97 41                      STAA   $41           ;   = any time when second is updated
22/    1009 : 97 43                      STAA   $43
23/    100B : 97 45                      STAA   $45
24/    100D :                     ;
25/    100D : 72 20 4B            CLCK10 OIM    #$20,$4B      ; Enable alarm interrupt
26/    1010 : BD FF A9                   JSR    SLEEP         ; MCU is sleep for save power
27/    1013 : 96 44                      LDAA   $44           ; Load "hour"
28/    1015 : 16                         TAB                  ; Display "hour"
29/    1016 : 84 F0                      ANDA   #$F0          ; (high order)
30/    1018 : 47                         ASRA
31/    1019 : 47                         ASRA
32/    101A : 47                         ASRA
```

```
33/     101B : 47                       ASRA
34/     101C : 8A 30                    ORAA   #'0'
35/     101E : CE 05 02                 LDX    #$0502
36/     1021 : 37                       PSHB
37/     1022 : BD FF 4C                 JSR    DSPLCH
38/     1025 : 32                       PULA                 ; Display (low order)
39/     1026 : 84 0F                    ANDA   #$0F
40/     1028 : 8A 30                    ORAA   #'0'
41/     102A : BD FF 4C                 JSR    DSPLCH
42/     102D : 86 3A                    LDAA   #':'          ;
43/     102F : BD FF 4C                 JSR    DSPLCH
44/     1032 : 96 42                    LDAA   $42           ; Load "minute"
45/     1034 : 16                       TAB                  ; Display "minute"
46/     1035 : 84 F0                    ANDA   #$F0          ; (high order)
47/     1037 : 47                       ASRA
48/     1038 : 47                       ASRA
49/     1039 : 47                       ASRA
50/     103A : 47                       ASRA
51/     103B : 8A 30                    ORAA   #'0'
52/     103D : CE 08 02                 LDX    #$0802
53/     1040 : 37                       PSHB
54/     1041 : BD FF 4C                 JSR    DSPLCH
55/     1044 : 32                       PULA                 ; Display (low order)
56/     1045 : 84 0F                    ANDA   #$0F
57/     1047 : 8A 30                    ORAA   #'0'
58/     1049 : BD FF 4C                 JSR    DSPLCH
59/     104C : 86 3A                    LDAA   #':'          ;
60/     104E : BD FF 4C                 JSR    DSPLCH
61/     1051 : 96 40                    LDAA   $40           ; Load "second"
62/     1053 : 16                       TAB                  ; Display "second"
63/     1054 : 84 F0                    ANDA   #$F0          ; (high order)
64/     1056 : 47                       ASRA
```

```
65/    1057 : 47                     ASRA
66/    1058 : 47                     ASRA
67/    1059 : 47                     ASRA
68/    105A : 8A 30                  ORAA   #'0'
69/    105C : CE 0B 02               LDX    #$0B02
70/    105F : 37                     PSHB
71/    1060 : BD FF 4C               JSR    DSPLCH
72/    1063 : 32                     PULA               ; Display (low order)
73/    1064 : 84 0F                  ANDA   #$0F
74/    1066 : 8A 30                  ORAA   #'0'
75/    1068 : BD FF 4C               JSR    DSPLCH
76/    106B : 20 A0                  BRA    CLCK10
77/    106D :                  ;
78/    106D :                        END
```

9.6.1 Binary dump format of object code

```
13 10 00 C6 00 BD FF 49 86 FF 97 41 97 43 97 45 72 20 4B BD FF A9 BD
13 10 13 96 44 16 84 F0 47 47 47 47 8A 30 CE 05 02 37 BD FF 4C 32 4A
14 10 26 84 0F 8A 30 BD FF 4C 86 3A BD FF 4C 96 42 16 84 F0 47 47 47 62
14 10 3A 47 8A 30 CE 08 02 37 BD FF 4C 32 84 0F 8A 30 BD FF 4C 86 3A 43
12 10 4E BD FF 4C 96 40 16 84 F0 47 47 47 47 8A 30 CE 0B 02 37 40
0D 10 60 BD FF 4C 32 84 0F 8A 30 BD FF 4C 20 A0 34
00 10 00 F0
```

Chapter 10

Floppy disk unit

10.1 General

The TF-20 Terminal Floppy is an intelligent floppy disk unit which is connected to the HX-20 through a serial communication interface and transfers the data stored in a floppy disk to the HX-20 according to the commands received from the HX-20.

When the TF-20 is connected to the HX-20, the `DBASIC.SYS` (Disk BASIC System, which is an extended portion of BASIC) is loaded from the floppy disk into the RAM of the HX-20 upon start of BASIC. The `DBASIC.SYS` loaded into the RAM operates together with the interpreter on the ROM until control is returned to the MENU again. It processes the data input/output to and from the floppy disk and newly added commands, statements and functions. The interpreter on the ROM handles the convetional functions of the HX-20.

In DISK BASIC, a maximum of two TF-20 units can be connected to the HX-20. The first TF-20 unit is used as disk drives "`A:`" and "`B:`" and the second unit as disk drives "`C:`" and "`D:`". To distinguish between the first and second units, the DIP switch located in the TF-20 must be used. The 4-pin DIP switch (bits 1 to 4) of the TF-20 is factory-set to all "ON" for drives "`A:`" and "`B:`". When connecting a second TF-20 unit to the HX-20, the DIP switch setting of the second unit must be changed to "bits 1, 2, 3, 4 = ON, ON, ON, OFF" to indicate that the unit is used as drives "`C:`" and "`D:`".

Daisy-chaining method is used to interconnect an HX-20 and a TF-20 or two TF-20 units via cable set #707 (for daisy chaining). TF-20 (disk 1) and TF-20 (disk 2) can be interconnected in any order. Figure 10.1 shows how two TF-20 units are connected to the HX-20.

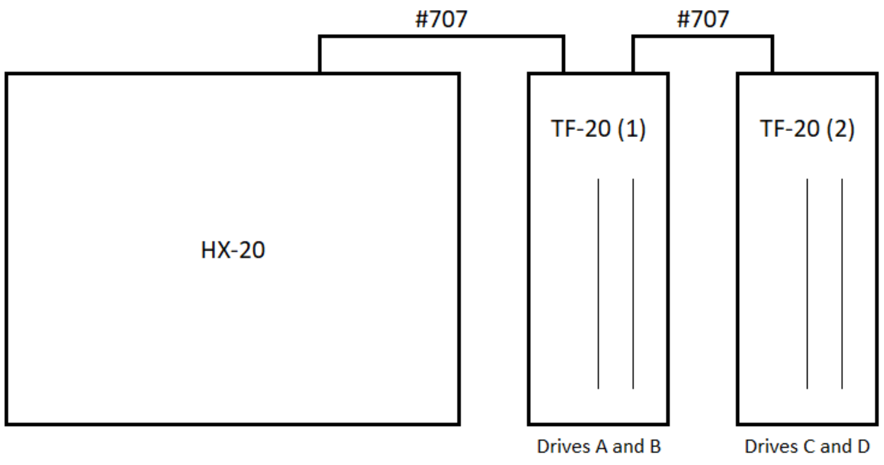


Figure 10.1: Interconnection of HX-20 and two TF-20 units

10.2 Disk format

Disk type:	double-sided, double density (MFM).
Number of tracks:	80 tracks (40 tracks × 2 sides).
Track density:	48 TPI.
Number of sectors:	16 sectors/track.
Capacity per sector:	256 bytes.
Total disk capacity:	320Kbytes (256 × 16 × 80).
Access time between tracks:	15ms.

Tracks and sectors are logically structured as shown below:

Number of tracks:	40 tracks (0 to 39).
Number of sectors:	64 sectors/track (1 to 64).
Capacity per sector:	128 bytes.

Table 10.3 shows the relationship between the physical and logical specifications.

	Physical specifications	Logical specifications
Track	One track on one side + one track on the other side.	One track.
Sector	One sector (256 bytes).	Two sectors (128 bytes × 2).

Table 10.3: Relationship between the physical and logical specifications.

All the floppy disks supplied by EPSON have been initialized before shipment so that they can be used as non-system disks. Floppy disks other than those supplied by EPSON and those disks in which a read or write error has occurred must be initialized by the `FRMAT` command. The system disk refers to the disk which contains a system program for DISK BASIC, and must be inserted into drive "`A:` " when DISK BASIC is to be booted. The system disk is mapped as follows:

Track 0	Sectors 1 and 2:	Cold-start loader (loads a system contained in the system disk into the memory of the TF-20).
	Sectors 3 to 18:	Unused.
	Sectors 19 to 46:	BDOS (Basic Disk Operating System).
	Sectors 47 to 64:	BIOS (Basic Input/Output System) for the HX-20.
Track 1	Sectors 1 to 42:	TFDOS (communication program with the HX-20).
	Sectors 43 to 64:	Unused.
Tracks 2 and 3	Sectors 1 to 64:	Unused.
Track 4	Sectors 1 to 16:	Directory area (for 64 directories max.)
	Sectors 17 to 64:	File area.
Tracks 5 to 38	Sectors 1 to 64:	File area (278Kbytes max.)

Two files "`BOOT80.SYS`" and "`DBASIC.SYS`" are secured for the system in the system disk. Since these files are write-protected, their filenames are not displayed even by executing the `FILES` command. Note that the user cannot use the same filenames as these two files. To duplicate a system disk, either copy all the contents of the existing system disk to a new floppy disk by `COPY` utility, or execute the `SYSGEN` command for a non-system disk.

"`SYSGEN`" copies not only the system area of the disk but also copies the system file whose file type is "`SYS`".

10.4 Interface with DISK BASIC

The DISK BASIC is broadly divided into the following 3 modules:

1. BASIC interpreter (ROM version: HX-20 side).
2. DBASIC interpreter (`DBASIC.SYS`: HX-20 side).

   This interpreter is an extended portion of BASIC which is loaded from a disk to the RAM of the HX-20 upon start of the BASIC and handles the data input/output to and from the disk and the processing of commands and statements, together with the BASIC interpreter described in 1 above. This module consists mainly of a portion connected to the BASIC interpreter (i.e., a BASIC driver) and a portion interfacing with the TFDOS (i.e., EPSP driver).
3. TFDOS (TF-20 side).

   The TFDOS which is resident on the RAM of the TF-20 receives commands from the HX-20. opens and reads or writes files using the BDOS or the BIOS for the HX-20, and returns data and error codes to the HX-20.

Of the above 3 modules, the BASIC driver and EPSP driver of the DBASIC interpreter are interfaced with each other through the `BSCINT` (BASIC interface), while the EPSP driver is interfaced with the TFDOS through the EPSP (EPSON Serial communication Protocol) as shown in Figure 10.2.

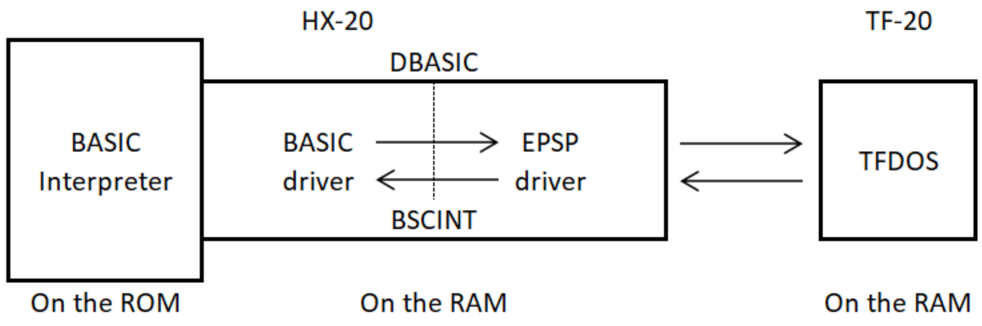


Figure 10.2: Software configuration of Disk BASIC

10.4.1 BASIC interface (`BSCINT`)

Functions of `BSCINT`

Interfacing of DBASIC with BASIC is supported by subroutine “`BSCINT`” (BASIC Interface) which has the following functions:

1. File open.
2. File close.
3. Random read (128 bytes).
4. Random write (128 bytes).
5. File delete.
6. File rename.
7. File size calculation.
8. First directory search.
9. Next directory search.
10. Direct write into disk (`DSKOS`, 128 bytes).
11. Disk formatting (`FRMAT`).
12. Disk system reset (`RESET`).
13. System disk generation (`SYSGEN`).
14. Disk free area calculation (`DSKF`).
15. Direct read from disk (128 bytes).
16. Disk all copy.

Subroutine call procedure

Subroutine "`BSCINT`" is called as follows:

1. Setting the entry point for `BSCINT`.

   The contents at an address 2 bytes from addresses (`0A3E` and `0A3F`) are "`JMP BSCINT`" (see Figure 10.3). This means that the address specified by addresses (`0A3E` and `0A3F`) is the entry point of the subroutine that includes `BSCINT` error processing.

2. Creation of a parameter packet.

   Parameters are created on memory, and are given in the order of the function code, return code, and data (see Figure 10.4). The data string has a length of one or more bytes. For details of the functions and parameters, refer to the `BSCINT` parameter table.

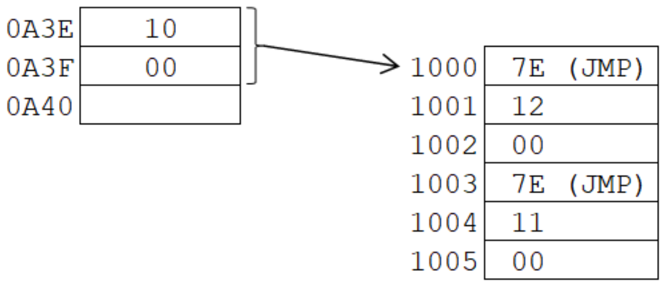


Figure 10.3: BSCINT entry point

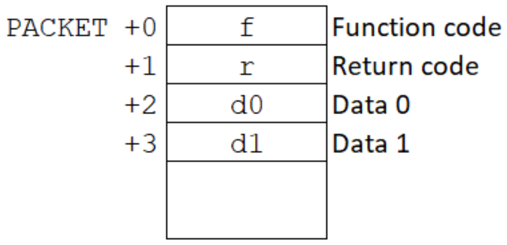


Figure 10.4: Parameter packet of subroutine BSCINT

3. Subroutine call.

   The first address of the parameter packet is set in the index register to call subroutine "BSCINT".

Example: file under the file descriptor "ABC.BAS" is opened in sequential output mode using drive "A".

```
       LDAA #$7E      ; (JMP instruction)
       STAA BSENTR
       LDD  $A3E
       STD  BSENTR+1
       LDX  #CPOPC
       JSR  BSENTR
       LDAA 1,X
       BNE  ERROR
       RTS
;      ...
ERROR  EQU  *         ; error procedure
;      ...
BSENTR FCB  $7E       ; (JMP BSCINT)
```

```
       RMB  2
CPOPC  FCB  $00
       FCB  $00
       FCB  $00
       FCC  "ABC     "
       FCC  "BAS"
```

10.4.2 BSCINT parameter packet table

All packet data numbers are decimal numbers.

No.	Function	Packet data No.	Description
1	File open		Opens the file in the specified drive according to the filename, file type, and file mode.
		00	`00` (function code).
		01	Return code (set at return).
		02	File number (set at return).
		03	Drive number (“`A`”, “`B`”, “`C`” or “`D`”).
		04-11	Filename (8 characters. If the filename is less then 8 characters, left-justify the filename and fill blank code(s) (`20`) in the remaining space).
		12-14	File type (3 characters. If the file type is less than 3 characters, left-justify the file type and fill blank codes (`20`) in the remaining space).

Continues in next page page...

...continued from previous page.

No.	Function	Packet data No.	Description
		15	Modes `10`$_{16}$: sequential input (`M.SQI`). `30`$_{16}$: sequential output (`M.SQO`). `40`$_{16}$: random access (`M.RND`). If no file exists in `M.SQI` or `M.SQO` mode, a new file is created. If no file exists in `M.SQI` mode, it is assumed that an error has occurred. If a file exists in `M.SQO` mode, the previous file will be deleted.
2	File close		Closes the specified opened file.
		00	`01` (function code).
		01	Return code (set at return).
		02	File number (i.e., the number returned at a file open).
3	Random read		Reads the specified record of a file. (One record consists of 128 bytes).
		00	`02` (function code).
		01	Return code (set at return).
		02	File number (i.e., the number returned at a file open).
		03-04	Record number (binary value in the range of 1 to 65535. Must be entered in the order of high- and low-order bytes).
		05-06	Buffer address (must be entered in the order of high- and low-order bytes).
4	Random write		Writes the specified record of a file. (One record consists of 128 bytes).
		00	`03` (function code).
		01	Return code (set at return).
		02	File number (i.e., the number returned at a file open).

Continues in next page page...

...continued from previous page.

No.	Function	Packet data No.	Description
		03-04	Record number (binary value in the range of 1 to 65535. Must be entered in the order of high- and low-order bytes).
		05-06	Buffer address (must be entered in the order of high- and low-order bytes).
5	File delete		Deletes the specified file.
		00	`04` (function code).
		01	Return code (set at return).
		02	Unused.
		03	Drive name (“`A`”, “`B`”, “`C`” or “`D`”).
		04-11	Filename (8 characters. If the filename is less then 8 characters, left-justify the filename and fill blank code(s) (`20`) in the remaining space).
		12-14	File type (3 characters. If the file type is less than 3 characters, left-justify the file type and fill blank codes (`20`) in the remaining space).
6	File rename		Rename the specified file.
		00	`05` (function code).
		01	Return code (set at return).
		02	Unused.
		03	Drive name (“`A`”, “`B`”, “`C`” or “`D`”).
		04-11	Filename before change (8 characters).
		12-14	File type before change (3 characters).
		15-22	Filename after change (8 characters).
		23-25	File type after change (3 characters).

Continues in next page page...

...continued from previous page.

No.	Function	Packet data No.	Description
7	File rename		Returns the number of records of the specified file. (One record consists of 128 bytes).
		00	`06` (function code).
		01	Return code (set at return).
		02	File number (i.e., the number returned at a file open).
		03-04	Maximum number of a record number (the number must be in the range of 0 to 65535. 0 indicates the null state).
8	First directory search		Returns the `FCB` (file control block) address and directory code on the disk of the file for which the filename and file type were specified. If the filename and file type are all specified by character '`?`', it is assumed that file matching has been completed for all files.
		00	`07` (function code).
		01	Return code (set at return).
		02	Unused.
		03	Drive name ("`A`", "`B`", "`C`" or "`D`").
		04-11	Filename (8 characters).
		12-14	File type (3 characters).
		15	Directory code (set at return).
		16-47	Directory `FCB` (set at return).
9	Next directory search		Searches the next directory. (This function is performed next to the function No. 8 above). The method of specifying the filename and file type is the same as function No. 8.
		00	`08` (function code).
		01	Return code (set at return).
		02	Unused.

Continues in next page page...

...continued from previous page.

No.	Function	Packet data No.	Description
		03	Drive name (“A”, “B”, “C” or “D”).
		04-11	Filename (8 characters).
		12-14	File type (3 characters).
		15	Directory code (set at return).
		16-47	Directory FCB (set at return).
10	Direct write into disk (DSKO$)		Writes data into the specified tracks and sectors of floppy disk.
		00	09 (function code).
		01	Return code (set at return).
		02	Unused.
		03	Drive name (“A”, “B”, “C” or “D”).
		04	Track number (binary value in the range of 0 to 39_{10}).
		05	Sector number (binary value in the range of 1 to 64_{10}).
		06-07	Buffer address (must be entered in the order of high- and low-order bytes).
11	Disk formatting (FRMAT)		Formats the floppy disk in the specified drive.
		00	0A (function code).
		01	Return code (set at return).
		02	Unused.
		03	Drive name (“A”, “B”, “C” or “D”).
12	Disk system reset		Enables disk replacement. When the disk system is reset, all the disks can be read or written and disk drive “A” is selected.
		00	0B (function code).
		01	Return code (set at return).
		02	Unused.
		03	Drive name (“A”, “B”, “C” or “D”).

Continues in next page page...

…continued from previous page.

No.	Function	Packet data No.	Description
13	System disk generation (SYSGEN)		Copies the system area and file of the system disk set in drive "A", to the disk set in drive "B". After copying, the disk in drive "B" can be used as a system disk.
		00	0C (function code).
		01	Return code (set at return).
14	Disk free area calculation (DSKF)		Provides the free area size of the disk in the specified drive in 2Kbyte units.
		00	0D (function code).
		01	Return code (set at return).
		02	Unused.
		03	Drive name ("A", "B", "C" or "D").
		04	Free area size (binary value in 2Kbyte units set at return).
15	Direct read from disk (DSKI$)		Reads data from the specified tracks and sectors of a floppy disk.
		00	0E (function code).
		01	Return code (set at return).
		02	Unused.
		03	Drive name ("A", "B", "C" or "D").
		04	Track number (binary value in the range of 0 to 39_{10}).
		05	Sector number (binary value in the range of 1 to 64_{10}).
		06-77	Buffer address (must be entered in the order of high- and low-order bytes. In this case, however, the message work area of EPSP driver routine is used).

Continues in next page page...

...continued from previous page.

No.	Function	Packet data No.	Description
16	Disk all copy		Copies all the contents of the floppy disk in the specified drive to the disk in the other drive of the same floppy disk unit (i.e., from "A" to "B, from "C" to "D").
		00	0F (function code).
		01	Return code (set at return).
		02	Unused.
		03	Drive name ("A", "B", "C" or "D"). **Note:** with drives "A" and "B", disk copying must be from "A" to "B"; with drives "C" and "D", disk copying must be from "C" to "D".

10.4.3 BSCINT return codes

Code (Hex.)	Meaning
00	Normal completion of operation.
01	The specified file is not found.
02	End of file (EOF) was detected during file input.
03	The file already exists.
04	The specified device is not found.
05	No directory area exists.
06	No disk area exists.
07	The specified record number is incorrect.
08	The disk is write-protected.
09	The file is not opened.
0A	The specified file number is incorrect.
0B	The specified file mode is incorrect.
0C	The specified file is already open.
0D	The number of opened files is too many.
0E	The specified file descriptor is incorrect.

Continues in next page page...

...continued from previous page.	
Code (Hex.)	Meaning
`0F`	An error has occurred during a read operation.
`10`	An error has occurred during a write operation.

10.5 EPSP (EPSON serial communication protocol)

10.5.1 EPSP functions

The EPSP is an interface between the EPSP driver and the TFDOS as described in Chapter 4. The EPSP on the TF-20 side has the following functions:

1. Disk system reset.

   Corresponds to item 12. of Subsection 10.4.2.

2. File open.

   Corresponds to item 1. of Subsection 10.4.2.

3. File close.

   Corresponds to item 2. of Subsection 10.4.2.

4. First directory search.

   Corresponds to item 8. of Subsection 10.4.2.

5. Next directory search.

   Corresponds to item 9. of Subsection 10.4.2.

6. File delete.

   Corresponds to item 5. of Subsection 10.4.2.

7. File creation.

   By this function, the directory and memory are initialized and a file empty of data is created.

8. Random read.

   Corresponds to item 3. of Subsection 10.4.2.

9. Random write.

   Corresponds to item 4. of Subsection 10.4.2.

10. File size calculation.

    Corresponds to item 7. of Subsection 10.4.2.

11. Disk all copy.

    Corresponds to item 16. of Subsection 10.4.2.

12. Direct write (128 bytes) into disk (`DSKO$`).

    Corresponds to item 10. of Subsection 10.4.2.

13. Disk formatting (`FRMAT`).

    Corresponds to item 11. of Subsection 10.4.2.

14. System disk generation (`SYSGEN`).

    Corresponds to item 13. of Subsection 10.4.2.

15. Disk free area calculation (`DSKF`).

    Corresponds to item 14. of Subsection 10.4.2.

16. Direct read (128 bytes) from disk (`DSKI$`).

    Corresponds to item 15. of Subsection 10.4.2.

17. Disk boot.

    By this function, file "`BOOT80.SYS`" is booted to the HX-20 from the system disk in the drive `A` of the TF-20. In other words, this function opens file "`BOOT80.SYS`", reads 128 bytes of data only and transfers them to the HX-20.

18. Load open.

    By this function, file "`DBASIC.SYS`" contained in the system disk in the drive `A` of the TF-20 is opened and then loaded into the RAM of the TF-20. After loading the file, the file is relocated on the RAM of the TF-20 using a relocatable flag (one of the load open parameters) and an ending or starting address. Return code "`FF`" if the corresponding file is not found, or return code "`00`" if found, is returned to the HX-20 together with the file size of "`DBASIC.SYS`".

19. Load close.

    This function indicates that the transfer of file "`DBASIC.SYS`" has been completed. In this case, the TF-20 does not perform any function.

20. Read one block.

    By this function, the file "`DBASIC.SYS`" opened, read and relocated in item 18 above is transferred to the HX-20 in units of 128 bytes.

    Return code "`FF`" indicates the end of file (`EOF`).

10.5.2 Subroutine "`OUTSRL`"

Subroutine "`OUTSRL`" handles the data transmission/reception of EPSP as follows:

1. Creation of a parameter packet.

   Parameters are given in the form of a packet as shown in Figure 10.5.

PACK	FMT	Text format 00: data transfer from the master (HX-20)
+1	DID	Terminal ID (drive A or B: 31_{16}; drive C or D: 32_{16})
+2	SID	Master ID (20_{16}) (HX-20)
+3	FNC	Message function
+4	SIZ	Text length minus 1
+5	d0	Data 0
	d1	Data 1
	...	

Figure 10.5: Parameter packet of subroutine `OUTSRL`

2. Subroutine call.

   The first address of the parameter packet is set in the index register to call subroutine "`OUTSRL`" (entry point: `FF70`). For details of the EPSP, refer to Chapter 4. For details of the EPSP functions on the TF-20 side, refer to Section 10.6.

   - EPSP side
     Open file
     Drive: "`A`"; filename, file type: `ABC`, `BAS`.
     File mode: sequential output "`O`".

```
OUTSRL EQU  $FF70
       LDX  #PACKET
       JSR  OUTSRL    ; Routine for data output
                      ; to the serial interface
;      ...
PACKET EQU  *
FMT    FCB  $00,$30,$30,$0F,$0E
MSG    FCB  $00,$01,$01
       FCC  "ABC     "
       FCC  "BAS"
```

10.6 Function table of floppy disk unit

FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
						Terminal floppy reset.
00	SS	MM	0E	00	00	XX
01	MM	SS	0E	00	00	Return code 00.
						File open.
00	SS	MM	0F	0E	00	High-order byte of FCB address in HX-20.
					01	Low-order byte of FCB address in HX-20.
					02	Drive code (1: drive A or 2: drive B).
					03-0A	Filename.
					0B-0D	File type.
					0E	Extent number (normally 0).
01	MM	SS	0F	00	00	Return code • BDOS error (see note at the end of this table). • FF: file not found. • Codes other than the above: normal.
						File close.

Continues in next page page...

...continued from previous page.

FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
00	SS	MM	10	01	00	High-order byte of FCB address in HX-20.
					01	Low-order byte of FCB address in HX-20.
01	MM	SS	10	00	00	Return code (the same return code as that at file open).
						First data search.
00	SS	MM	11	0C	00	Drive code (1 or 2).
					01-08	Filename.
					09-0B	File type.
					0C	Extent number (normally 0).
01	MM	SS	11	20	00	Return code (the same return code as that at file open).
					01-20	Directory FCB entry (the FCB of the found directory is entered).
						Next data search.
00	SS	MM	12	00	00	XX.
01	MM	SS	12	20	00	Return code (the same return code as that at file open).
					01-20	Directory FCB entry (the FCB of the found directory is entered).
						File creation.
00	SS	MM	16	0E	00	High-order byte of FCB address in HX-20.
					01	Low-order byte of FCB address in HX-20.
					02	Drive code (1 or 2).
					03-0A	Filename.
					0B-0D	File type.
					0E	Extent number (normally 0).
01	MM	SS	16	00	00	Return code (the same return code as that at file open).
						File rename.
00	SS	MM	17	1F	00	Drive code (1 or 2).
					01-08	Filename before change (8 characters).

Continues in next page page...

...continued from previous page.

FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
					09-0B	File type before change (3 characters).
					0C	Extent number.
					0D-0F	Unused.
					10	Drive code (1 or 2).
					11-18	Filename after change (8 characters).
					19-1B	File type after change (3 characters).
					1C	Extent number.
					1D-1F	Unused.
01	MM	SS	17	00	00	Return code (the same return code as that at file open).
						Random data read.
00	SS	MM	21	04	00	High-order byte of FCB address in HX-20.
					01	Low-order byte of FCB address in HX-20.
					02	R0 } Random record numbers
					03	R1 } Random record numbers
					04	R2 } Random record numbers
01	MM	SS	21	82	00	Extent number.
					01	Current record number.
					02-81	Read data (128 bytes).
					82	Error code • BDOS error (see note at the end of this table). • Codes other than the above: normal.
						Random data write.
00	SS	MM	22	84	00	High-order byte of FCB address in HX-20.
					01	Low-order byte of FCB address in HX-20.

Continues in next page page...

...continued from previous page.

FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
					02-81	Write data (128 bytes).
					82	R0 } Random record numbers
					83	R1 } Random record numbers
					84	R2 } Random record numbers
01	MM	SS	22	02	00	Extent number.
					01	Current record number.
					02	Error code • BDOS error (see note at the end of this table). • Codes other than the above: normal.
						File size calculation.
00	SS	MM	23	01	00	High-order byte of FCB address in HX-20.
					01	Low-order byte of FCB address in HX-20.
01	MM	SS	23	05	00	Extent number.
					01	Current record number.
					02	R0 } Random record numbers
					03	R1 } Random record numbers
					04	R2 } Random record numbers
					05	Return code (always 0).
						Disk all copy.
00	SS	MM	7A	00	00	Drive code (1 or 2).
01	MM	SS	7A	02	00	High-order byte of currently copied track number.
					01	Low-order byte of currently copied track number. • 0 to 39. • FFFF: end.
					02	Return code (BDOS error or 0).

Continues in next page page...

...continued from previous page.

FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
						Direct write into disk.
00	SS	MM	7B	82	00	Drive code (1 or 2).
					01	Track number (0 to 39).
					02	Sector number (1 to 64).
					03-82	Write data (128 bytes).
01	MM	SS	7B	00	00	Return code (BDOS error or 0).
						Disk formatting (FRMAT).
00	SS	MM	7C	00	00	Drive code (1 or 2).
01	MM	SS	7C	02	00	High-order byte of currently formatted track number.
					01	Low-order byte of currently formatted track number. • 0 to 39. • FFFF: end.
					02	Return code (BDOS error or 0).
						New system disk generation (SYSGEN).
00	SS	MM	7D	00	00	XX.
01	MM	SS	7D	02	00-01	• 0000: not end. • FFFF: end.
					02	Return code (BDOS error or 0).
						Disk free area calculation (DSKF).
00	SS	MM	7E	00	00	Drive code (1 or 2).
01	MM	SS	7E	02	00	Free area size (in 2Kbyte units).
					01	Return code (BDOS error or 0).
						Direct read from disk (DSKI$).
00	SS	MM	7F	02	00	Drive code (1 or 2).
					01	Track number (0 to 39).
					02	Sector number (1 to 64).

Continues in next page page...

...continued from previous page.

FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
01	MM	SS	7F	80	00-7F	Read data (128 bytes).
					80	Return code (BDOS error or 0).
						Disk boot.
00	SS	MM	80	00	00	Application ID (in BASIC 80_{16} ... BOOT80.SYS).
01	MM	SS	80	FF	00	Return code • 00: normal. • FF: file not found.
					01-FF	Read data.
						Load open.
00	SS	MM	81	0D	00-07	Filename (the filename of DISK BASIC is “DBASIC”).
					08-0A	File type (the file type of DISK BASIC is “SYS”).
					0B	Relocate flag • 00: do not relocate. • 01: relocate from the starting address. • 02: relocate from the ending address.
					0C-0D	Ending or starting address.
01	MM	SS	81	02	00	Return code • 00: normal. • FF: file not found.
					01	High-order byte of the file size.
					02	Low-order byte of the file size.
						Load close.
00	SS	MM	82	00	00	XX.

Continues in next page page...

. . . continued from previous page.

FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
01	MM	SS	82	00	00	Return code (always 0).
						Read one block.
00	SS	MM	83	01	00	High-order byte of current record number.
					01	Low-order byte of current record number.
01	MM	SS	83	82	00	High-order byte of current record number.
					01	Low-order byte of current record number.
					02-81	Read data.
					82	Return code (00; normal; FF: end).

Note: the term "BDOS error" used in the above table refers to one of the following errors:

- FA: read error.
- FB: write error.
- FC: drive select error.
- FD or FE: write protect error.

The format of the file control block (FCB) used by the floppy disk unit is as follows:

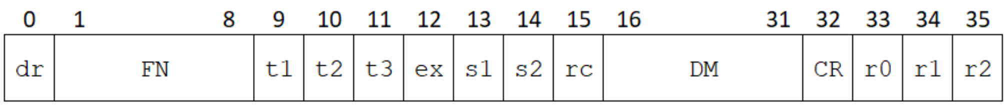


Figure 10.6: FCB format

- dr: disk drive code (00 to 16) (use of code 05 to 16 wil result in an error).
  - 00: a file is assigned to the standard disk drive.
  - 01: disk and disk drive A are selected automatically.

  - `02`: disk and disk drive `B` are selected automatically.

    ...

  - `16`: disk and disk drive `P` are selected automatically.

- `FN`: filename consisting of a maximum of 8 characters (in ASCII codes).

  If no filename is given by the user, blanks (`20`) will be filled as the filename.

- `t1`, `t2`, `t3`: file format (in ASCII codes).

  As ASCII codes, bits in the upper row are selected and high-order bits set to 0 are used. These bits when represented by `t1`, `t2` and `t3` are as follows:

  - `t1`=1: read only file.
  - `t2`=1: no system file, `FILES` list.

- `ex`: file extent (normally `0`).

  This is a number to indicate the current location of the logical extent, and is normally set to `00` by the user. This number must be a value in the range of 0 to 31 when a file input/output is to be performed.

- `s1`: used within the system.

- `s2`: used within the system. `s2` is set to 0 when a file is to be opened, created or called for search.

- `rc`: record number of the logical extent indicated by "`ex`" and must be a value in the range of 0 to 128.

- `DM`: a value set and used by the system.

- `cr`: a value indicating the location of the record where data read/write is being performed in sequential file processing. This value is normally set to `0` by the user.

- `r0`, `r1`, `r2`: random record number indicated by a value in the range of 0 to 65535. `r0`, `r1` and `r2` are used to configure 24 bits. `r0` indicates the low-order digit, `r1` the high-order digit and `r2` an overflow.

Chapter 11

Slave MCU commands

11.1 General

The interface between the master and slave MCUs consists of two signal lines. Serial communication is performed at 38.4Kbps. Slave MCU operations are performed in response to instructions (commands) sent from the main MCU. The master CPU uses the serial interface to communicate either with the slave MCU or externally.

The slave CPU supports the following functions:

1. Operation of the microprinter.
2. Data reception via RS-232C port.
3. Data I/O for external cassette.
4. Data I/O and operation of the built-in microcassette.
5. Output for piezoelectric speaker.
6. Control switches for serial, power supply and bar code reader power.

11.2 Commands for slave MCU control

Commands are sent to the slave MCU via the 38.4Kbps serial interface. Commands are one byte in length. However, for some commands, parameters are added. The standard communication procedure involves sending a command from the master MCU and receiving an `ACK` signal from the slave MCU in response. The sequence for commands sent with parameters is shown below.

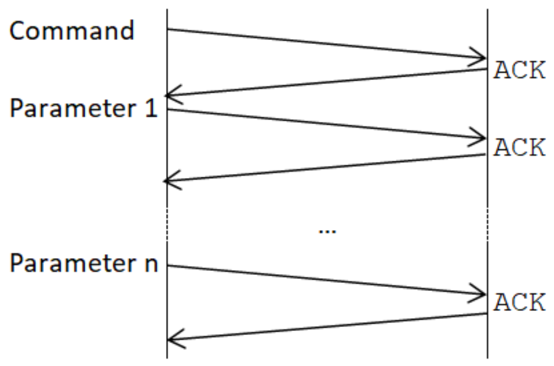


Figure 11.1: `FCB` format

First, a 1-byte command is sent to the slave MCU. The `SNSCOM` subroutine (entry point `FF19`) is called to receive the `ACK` signal. For details of commands, see the command table.

For data reception from the RS-232C or cassette, the slave MCU sends serial input data to main MCU upon completion of command reception. Data received by the slave MCU under this condition are assumed to be commands and the current input mode is cancelled.

11.3 Cancelling a command

The command being executed is cancelled if an overrun occurs during serial communication. (For example, if overrun occurs when 100-line feed is specified for the microprinter, the current command is aborted and the system goes into `WAIT` status pending receipt of a fresh command). If new data is received from the main MCU while a command is being executed by the slave MCU, the data is set in the receive register but not processed. At this point, if new serial communication data is received, the data in the register is destroyed, causing an overrun error.

To cancel a command, the master MCU sends a series of `BREAK` commands to the slave MCU. Subroutine `BREAKIO` (entry point `FFA3`) is provided for this purpose.

11.4 Slave MCU command transmission subroutine

Subroutine name	Entry point	Description
`SNSCOM`	`FF19`	Transfers a command or 1 byte of data to the slave MCU via the SCI. • Parameters – At entry ∗ `(A)`: transmit data (command). – At return ∗ `(C)`: abnormal I/O flag. ∗ `(A)`: return code (transmit data from slave MCU). • Registers retained: `(B)`, `(X)`. • Subroutines referenced: none. • Variables used: none.

11.5 Commands to slave MCU

Command	Master MCU data	Slave MCU response	Description
`00`	`00`	`01` (`ACK`)	Slave MCU ready check. `ACK` is returned when the slave MCU is ready to receive a command. The slave MCU makes no response if it is not ready.
`01`	`01`	`01` (`ACK`)	Sets the constants required by slave MCU in the field. The following values are set: generated polynomial expressions, `BCC` register value, RS-232C bit rate, cassette (external or built-in microcassette), microcassette tape counter setting.

Continues in next page...

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
02	02	01 (ACK)	Inizialization. The status of serial communication driver remains unchanged.
03	03 (command) AA (parameter)	01 (ACK)	Opens mask for special commands. Commands 06, 07, 08 and 0B cannot be executed unless the masks are opened. Any value other than AA indicates that the mask is closed.
04	04	01 (ACK)	Closes masks for special commands.
05	05 ah (upper byte of address) al (lower byte of address)	01 (ACK) 01 (ACK) d (data)	Reads slave MCU memory. NAK (0F) is returned in response to 05 if the mask is not open.
06	06 ah (upper byte of address) al (lower byte of address) d (data)	01 (ACK) 01 (ACK) 01 (ACK) 01 (ACK)	Stores data to the memory address specified by the slave MCU. NAK (0F) is returned and command execution is aborted if the mask is not open.
07	07 ah (upper byte of address) al (lower byte of address) d (data)	01 (ACK) 01 (ACK) 01 (ACK) 01 (ACK)	Performs logical OR operation for the data at the memory address specified by the slave MCU and the specified data and stores the result in the specified address. 0F (NAK) is returned and command execution is aborted if the mask is not opened.

Continues in next page...

...continued from previous page.

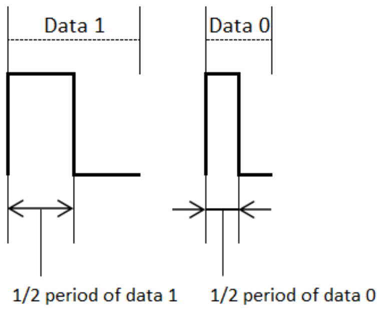
Command	Master MCU data	Slave MCU response	Description
`08`	`08` `ah` (upper byte of address) `al` (lower byte of address) `d` (data)	`01` (`ACK`) `01` (`ACK`) `01` (`ACK`) `01` (`ACK`)	Performs logical `AND` operation for the data at the memory address specified by the slave MCU and the specified data and stores the result in the specified address. `0F` (`NAK`) is returned and command execution is aborted if the mask is not opened.
`09`	`09`	`01` (`ACK`)	Unused (in version 2, bar-code reader power ON).
`0A`	`0A`	`01` (`ACK`)	Unused (in version 2, bar-code reader power OFF).
`0B`	`0B` `ah` (upper byte of address) `al` (lower byte of address)	`01` (`ACK`) `01` (`ACK`) `01` (`ACK`)	Sets the program counter to a specified value (jumps execution to a specified address). `0F` (`NAK`) is returned and command execution is aborted if the mask is not opened.
`0C`	`0C`	`02` (`ACK` for `BREAK`)	`BREAK`. Terminates processing and sets the system to command `WAIT` status.

`0D`	`0D`
	`AA`
		`01` (`ACK`)
		`01` (`ACK`)
			Cuts OFF power supply.
			Command execution is aborted if parameter `AA` is omitted.

`0E-0F`			Undefined.
`10`	`10` `d` (data)	`01` (`ACK`) `01` (`ACK`)	Activates the built-in printer. Prints out 6-dot data (bit 0 to bit 5). One dot-line is printed by repeating this command procedure 24 times.

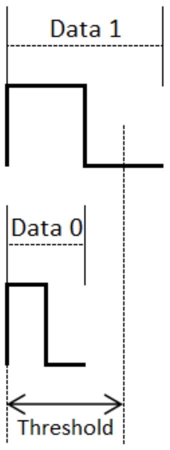
Continues in next page...

…continued from previous page.

Command	Master MCU data	Slave MCU response	Description
11	11 d (number of lines)	01 (ACK) 01 (ACK)	Feeds the specified number of dot lines to the built-in printer.
12	12	01 (ACK)	Activates built-in printer motor for approx. 1.2s (paper feed operation).
13-1F			Undefined.
20	20	21 (ACK)	Executes external cassette ready check. Code 21 is returned when the external cassette is ready. The external cassette makes no response if it is not ready.
21	21 d1 (upper byte of time (MCU clocks) of 1/2 cycle for data '1') d2 (lower byte of time (MCU clocks) of 1/2 cycle for data '1') d3 (upper byte of time (MCU clocks) of 1/2 cycle for data '0') d4 (lower byte of time (MCU clocks) of 1/2 cycle for data '0')	01 (ACK) 21 (ACK) 21 (ACK) 21 (ACK) 21 (ACK)	Sets constants for the external cassette.  Data 1 Data 0 1/2 period of data 1 1/2 period of data 0 The times (in MCU clock pulses) for 1/2 cycle for data '1' and for data '0' are set as constants. The bit judgement threshold value for data read is also set as the number of MCU clocks.

Continues in next page…

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
	`d5` (upper byte of bit judgement threshold value between cycle times for '`1`' and '`0`')	`21` (`ACK`)	
	`d6` (lower byte of bit judgement threshold value)	`21` (`ACK`)	
	`d7` (upper byte of interblock gap length (in bytes) in stop mode (tape head stops between blocks))	`21` (`ACK`)	This data represents the interblock gap length in tape stop mode (long gap) as the number of times that data `FF` is written to the tape.
	`d8` (upper byte of interblock gap length in stop mode)	`21` (`ACK`)	
`22`	`22`	`01` (`ACK`)	Turns the external cassette REM terminal ON.
`23`	`23`	`01` (`ACK`)	Turns the external cassette REM terminal OFF.
`24`	`24`	`01` (`ACK`)	Writes 1 block of data in EPSON format.

Continues in next page...

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
	`d1` (block write start mode)	`21` (`ACK`)	After synchronizing pattern is sent, the number of bytes specified as the block length is written followed by 2 CRC bytes.
	`d2` (block write end mode)	`21` (`ACK`)	For output data, only the number of bytes specified as the block length are required. If data has not been received from the master MCU when the slave attempts to write data to the cassette, the slave MCU returns `2F`, activates the speaker (880Hz for 1s) and terminates cassette output.
	`d3` (upper byte of block length)	`21` (`ACK`)	Block write start mode values are as follows (`d1`):
	`d4` (lower byte of block length)	`21` (`ACK`)	`00`: 125-byte gap before the block (default value).
	`W1` (output data) ...	`22` (`ACK`) (`2F` (`NAK`))	`01`: 15-byte gap before the block. `FF`: 625-byte gap before the block.
	`W`*n* (output data)	`22` (`ACK`) (`2F` (`NAK`))	Block write start mode value (`00`, `01` or `FF`) is used as the block write end mode value at the completion of block write operation. In `00` and `FF` modes, the REM terminal is turned after completion of block write operation.
`25`	`25`	`01` (`ACK`)	

Continues in next page...

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
	`d1` (upper byte of number of `FF` patterns)	`21` (`ACK`)	Outputs number of `FF` patterns specified by `d1` and `d2` to the external cassette.
	`d2` (lower byte of number of `FF` patterns)	`21` (`ACK`)	Writing of data `FF` is unrelated to blocking.
`26`	`26`	`01` (`ACK`)	Inputs files from an external cassette.
	`d1` (block read start mode)	`21` (`ACK`)	Searches header block (EPSON format) and sends the contents of this block to the master MCU.
	`d2` (block read end mode)	`21` (`ACK`)	Header block always begins with data `H`. In actual practice, however, `d1` is ignored.
	`d3` (upper byte of block length)	`21` (`ACK`)	REM is turned OFF after reading 1 block if `d2` is `00`.
	`d4` (lower byte of block length)	`21` (`ACK`)	If `d2` is `01`, REM is left ON.
		`W1` `W2` `W3` ... `W84`	If an error occurs during transmission of block data, data transmission is terminated and `P34` (connected to `P12` of the master MCU) is turned ON. Two CRC bytes are placed at the end of the block but are not transmitted.
`27`	`27`	`01` (`ACK`)	Inputs files from an external cassette.

Continues in next page...

…continued from previous page.

Command	Master MCU data	Slave MCU response	Description
	`d1` (block read start mode)	`21` (`ACK`)	Searches `EOF` block (EPSON format) and sends the contents of this block to the master MCU.
	`d2` (block read end mode)	`21` (`ACK`)	Header block always begins with data `E`.
	`d3` (upper byte of block length)	`21` (`ACK`)	Parameters and execution result are identical to those for command `26`.
	`d4` (lower byte of block length)	`21` (`ACK`)	

		`W1`
		`W2`
		`W3`
		…
		`W84`	

`28`	`28`	`01` (`ACK`)	Inputs files from an external cassette.
	`d1` (block read start mode)	`21` (`ACK`)	Inputs the next block (EPSON format) and sends the data to the master MCU.
	`d2` (block read end mode)	`21` (`ACK`)	The block may begin with any data.
	`d3` (upper byte of block length)	`21` (`ACK`)	Parameters and execution result are identical to those for command `26`.
	`d4` (lower byte of block length)	`21` (`ACK`)	

Continues in next page...

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
		W1 W2 W3 ... W260	
29-2A			Undefined.
2B	2B	01 (ACK)	Specifies the input signal for the external cassette and built-in microcassette.

Continues in next page...

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
	d1 (specifies the pulse mode)	`21` (`ACK`)	`d1` meaning: • Bit 3: when logic '`1`', the microcassette input signal is as defined by bit 2. When logic '`0`', the microcassette input signal is judged at input. • Bit 2: when logic '`1`', the microcassette input signal is reversed. When logic '`0`', the microcassette input signal is normal. • Bit 1: when logic '`1`', the external cassette input signal is as defined by bit 0. When logic '`0`', the external cassette input signal is judged at input. • Bit 0: when logic '`1`', the external cassette input signal is reversed. When logic '`0`', the external cassette input signal is normal. **Note:** in versions 1 and 2, the slave MCU assumes (bit 3, bit 2) = $(1, 1)$ when bit 3 is logic '`0`'.

Continues in next page...

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
`30`	`30` `d1` (tone) `d2` (duration)	`01` (`ACK`)	Specifies the tone and duration and sounds the piezoelectric speaker. The specifications for tone are as follows: 0 = pause, 1, 2, 3,... correspond to C, D, E,... Values 1 to 28_{10} represent a 4-octave major scale (13 = 880Hz) and values 29 to 56_{10} a scale each tone of which is a half tone higher than that represented by 1 to 28. Duration is specified with 1 = 0.1s, 2 = 0.2s, etc. 0 specifies a pause (command not executed).
`31`	`31`	`01` (`ACK`)	Specifies the frequency and duration and sounds the piezoelectric speaker.
	`d1` (upper byte of frequency specification)	`31` (`ACK`)	Frequency is specified as the number of MCU clocks corresponding to 1/2 cycle.
	`d2` (lower byte of frequency specification)	`31` (`ACK`)	Example: 349_{10} for 880Hz.
	`d3` (upper byte of block duration specification)	`31` (`ACK`)	Specification of duration: 1 = 400µs (256 MCU clocks).
	`d4` (lower byte of block duration specification)	`31` (`ACK`)	
`32`	`32`	`01` (`ACK`)	Sounds the speaker for 0.03s at tone 6 using command `30`.

Continues in next page...

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
33	33	01 (ACK)	Sounds the speaker for 1s at tone 20 using command 30.
34	34 d s1 d l1 d s2 d l2 ... d sn d ln FF	01 (ACK) 31 (ACK) 31 (ACK) 31 (ACK) 31 (ACK) ... 31 (ACK) 31 (ACK) 31 (ACK)	Sets melody data in the slave MCU buffer. Buffer size is 48 bytes. The data set here can be output to the speaker using command 35. The format for data is the same as for command 30. i.e., tone, duration. As a pair, these data repeatedly specify the tone and duration. Due to the buffer size, the maximum number of data is 46_{10}. Data must end with FF. The data set in the buffer remains unchanged unless it is rewritten by command 34 or destroyed by a printer command (this is because this buffer is also used by printer).
35	35	01 (ACK)	Sounds the piezoelectric speaker with the melody data specified in command 34.
36-3F			Undefined.
40	40	01 (ACK)	Turns the serial driver ON. RTS is set to low (OFF).
41	41	01 (ACK)	Turns the serial driver OFF.
42	42 d1 (upper byte of bit rate) d2 (lower byte of bit rate)	01 (ACK) 41 (ACK) 41 (ACK)	Selects RS-232C mode. Bit rate corresponds to bit time specified as the number of CPU clock cycles. For example, 800_{16}: 300bps.

Continues in next page...

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
	`d3` (word length)	`41` (`ACK`)	Word length (excluding parity bits) may be set at 5, 6, 7 or 8 bits.
	`d4` (mode)	`41` (`ACK`)	The significance of each bit of mode data (`d4`) is as follows: • Bits 0, 1: number of stop bits (1 or 2). • Bit 2: '`0`', carrier check; '`1`', no carrier check. • Bit 3: controls `RTS` output ('`0`': low; '`1`': high. • Bits 4, 5: undefined. • Bits 6, 7: parity control ('`00`': even parity; '`01`': odd parity; '`10`' or '`11`': none).

Continues in next page...

…continued from previous page.

Command	Master MCU data	Slave MCU response	Description
`43`	`43`	`V`	Inputs RS-232C status maintained by the slave MCU. The significance of each bit in the status code is as follows: • Bit 0: carrier detect. • Bit 1: parity. • Bit 2: overrun. • Bit 3: framing. • Bits 4-7: undefined. Error status bits are reset by a clear command (`44`) or when input is resumed (command `45`).
`44`	`44`	`01` (`ACK`)	Clears RS-232C error status.

`44`	`44`	`01` (`ACK`)
		`V1`
		`V2`
			Starts RS-232C input. Input data is sent to the master MCU. If the word length of the data (including the parity bits) is less than 8 bits, the remaining bits (from MSB) are padded with `0` (right-justified).
			`P34` (connected to master MCU `P12`) is reset (logic '`1`') when input starts.
			`P34` is set (logic '`1`') if an error (framing error, carrier OFF, etc.) occurs.
			Data reception terminates upon receipt of a new command from the master MCU.

Continues in next page…

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
46	46	01 (ACK)	Terminates RS-232C input initiated by command 45 (this is not the only way of terminating such input).
47			Undefined.
48	48 d1 (upper byte of polynomial expression). d2 (lower byte of polynomial expression).	01 (ACK) 41 (ACK) 41 (ACK)	Sets the polynomial expression used for CRC check. This polynomial expression can also be used for cassette files. Default value is 8408 ($1+x^5+x^{12}+x^{16}$).
49	49 d1 (upper byte of BCC register value). d2 (lower byte of BCC register value).	01 (ACK) 41 (ACK) 41 (ACK)	Sets BCC register values for CRC check. This value is used as the initial value when CRC calculation is performed at RS-232C input. However, the data in BCC register is lost when I/O operations to a cassette are performed.
4A	4A	V	Inputs upper byte of BCC value.
4B	4B	V	Inputs lower byte of BCC value.
4C	4C	01 (ACK)	Activates the serial driver. In contrast to command 40 which turns RTS OFF, this command does not affect the status of RTS.

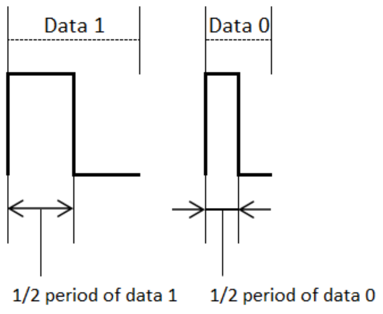
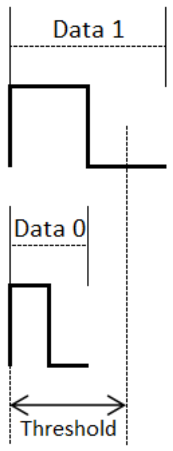
Continues in next page...

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
4D	4D	01 (ACK)	RTS high/low specification. Only bit 0 is significant. 0: low; 1: high.
4E-4F			Undefined.
50	50	V	Identified the plug-in option. Bit states of P46 and P20 are returned. • Bit 0: bit state of P46. • Bit 1: bit state of P20. • bits 2 to 7: 0. **Note:** plug-in option power is turned OFF when this command is executed.
51	51	01 (ACK)	Turns power of plug-in ROM cartridge ON.
52	52	01 (ACK)	Turns power of plug-in ROM cartridge OFF.
53-5F			Undefined.
60	60	61 (ACK)	Executes ready check. Slave MCU responds only if a microcassette command is executable. In all other cases, no response is sent.
61	61 d1 (upper byte of signal low time of 1 cycle for data '1')	01 (ACK) 61 (ACK)	Sets the microcassette parameters. Parameters are specified using data d1 to d8.

Continues in next page...

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
	`d2` (lower byte of signal low time of 1 cycle for data '1')	`61` (`ACK`)	 Data 1 Data 0 1/2 period of data 1 1/2 period of data 0
	`d3` (upper byte of signal high time of 1 cycle for data '1')	`61` (`ACK`)	
	`d4` (lower byte of signal high time of 1 cycle for data '1')	`61` (`ACK`)	
	`d5` (upper byte of time of 1/2 cycle for data '0')	`61` (`ACK`)	
	`d6` (lower byte of time of 1/2 cycle for data '0')	`61` (`ACK`)	
	`d7` (upper byte of '0', '1' bit judgement threshold value)	`61` (`ACK`)	
	`d8` (lower byte of `d7`)	`61` (`ACK`)	
`62`	`62` `d1` (upper byte of number of gap bytes)	`01` (`ACK`) `61` (`ACK`)	Specified the number of gap bytes for each mode when stopping the microcassette between blocks.

Continues in next page...

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
	`d2` (lower byte of number of gap bytes)	`61` (`ACK`)	
`63`	`63`	`01` (`ACK`)	Advances the tape (in PLAY mode) for the specified number of bytes.
	`d1` (upper byte of number of bytes sent)	`61` (`ACK`)	The bit judgement threshold value is taken as the length of one bit and 9 bits are counted as one byte.
	`d2` (lower byte of number of bytes sent)	`61` (`ACK`)	This command does not perform data read.
`64`	`64`	`01` (`ACK`)	
	`d1` (block write start mode)	`61` (`ACK`)	Outputs one block to microcassette in EPSON format.
	`d2` (block write end mode)	`61` (`ACK`)	Output file and command format and execution result are identical to command `24` (block output to external cassette).
	`d3` (upper byte of block length)	`61` (`ACK`)	
	`d4` (lower byte of block length)	`61` (`ACK`)	
	`W1` (data)	`61` (`ACK`) (`6F` (`NAK`))	
	...		
	`W`n (data)	`61` (`ACK`)	
`65`	`65`	`01` (`ACK`)	

Continues in next page...

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
	d1 (upper byte of number of bytes) d2 (lower byte of number of bytes)	61 (ACK) 61 (ACK)	Outputs the number of bytes of data FF specified by d1 and d2 to the microcassette. Result is the same as command 25.
66	66 d1 (block read start mode) d2 (block read end mode) d3 (upper byte of block length) d4 (lower byte of block length)	01 (ACK) 61 (ACK) 61 (ACK) 61 (ACK) 61 (ACK) W1 W2 ... W84	Inputs files from microcassette. Command operation and parameters are identical to command 26.
67	67 d1 (block read start mode) d2 (block read end mode) d3 (upper byte of block length)	01 (ACK) 61 (ACK) 61 (ACK) 61 (ACK)	Inputs files from microcassette. Command operation and parameters are identical to command 27.

Continues in next page...

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
	d4 (lower byte of block length)	61 (ACK)	
		W1 W2 ... W84	
68	68	01 (ACK)	Inputs files from microcassette.
	d1 (block read start mode)	61 (ACK)	Command operation and parameters are identical to command 28.
	d2 (block read end mode)	61 (ACK)	
	d3 (upper byte of block length)	61 (ACK)	
	d4 (lower byte of block length)	61 (ACK)	
		W1 W2 ... W260	
69-6C			Undefined.
6D	6D	01 (ACK)	
	d1 (upper byte of counter value)	61 (ACK)	Sets microcassette counter value in the slave MCU.
	d2 (lower byte of counter value)	61 (ACK)	The counter value is a 16-bit signed hexadecimal number.

Continues in next page...

…continued from previous page.

Command	Master MCU data	Slave MCU response	Description
6E	6E	V	Fetches microcassette counter value. Sends the upper 8 bits of counter value to the master MCU.
6F	6F	V	Fetches microcassette counter value. Sends the lower 8 bits of counter value to the master MCU.
70	70	V	Executes microcassette write protect check. In write enable status, '0' is returned to the master MCU. In write protect status, 'FF' is returned to MCU.
71	71 d1 (upper byte of counter value) d2 (lower byte of counter value)	01 (ACK) 61 (ACK) 61 (ACK)	Rewinds microcassette to the tape counter value specified by d1 and d2. Speed of rewind is the same as that of fast forward.
72	72 d1 (upper byte of counter value) d2 (lower byte of counter value)	01 (ACK) 61 (ACK) 61 (ACK)	Advances the microcassette tape (fast forward) to the tape counter value specified by d1 and d2.

Continues in next page...

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
`73`	`73`	`01` (`ACK`)	Causes the microcassette to rewind up to the beginning of tape (fast rewind).
`74`	`74`	`V`	Inputs microcassette status to the slave MCU. Status is a one-byte code. The significance of each bit is as follows (logic '`1`' indicates an error): • Bit 0: tape read error. • Bit 1: undefined. • Bit 2: header or `EOF` block not found. • Bit 3: delay in data transmission from master MCU during data output. • Bit 4: write protect. • Bit 5: head error. • Bit 6: microcassette not connected. • Bit 7: undefined.
`75`	`75`	`01` (`ACK`)	Clears the microcassette status register.
`76`	`76`	`01` (`ACK`) (`6F` (`NAK`))	Loads the microcassette head. If an error occurs during loading, the slave MCU returns '`6F`'.

Continues in next page...

...continued from previous page.

Command	Master MCU data	Slave MCU response	Description
77	77	01 (ACK) (6F (NAK))	Unloads the microcassette head. If an error occurs during unloading, the slave MCU returns '6F'.
78	78	01 (ACK) (6F (NAK))	Rewinds the microcassette tape. Rewind operation continues until the next command is received.
7A	7A	01 (ACK) (6F (NAK))	Advances the microcassette tape (slow forward). Slow forward continues until the next command is received.
7B	7B	01 (ACK) (6F (NAK))	Stops microcassette tape forward and rewind operations.
7C-7F			Undefined.
80	80	01 (ACK) (0F (NAK))	Causes master MCU PLG2 port (address 26, bit 5) value to be stored in the specified bit in the slave MCU.
	d1 (upper byte of address)	01 (ACK)	The PLG2 port value is stored in the bit specified by d3 at the slave MCU address dpecified by d1 and d2.
	d2 (lower byte of address)	01 (ACK)	This operation continues until the next command is received.
	d3 (bit position)	01 (ACK)	As this is a special command, the mask must be opened prior to execution (command 03). This command will not be accepted if the mask has not been opened.
81	81	01 (ACK) (0F (NAK))	Stores the value of the specified bit in the slave MCU to P12 of the master MCU.

Continues in next page...

...continued from previous page.			
Command	Master MCU data	Slave MCU response	Description
	`d1` (upper byte of address)	`01` (`ACK`)	The slave MCU address is specified by `d1` and `d2`, and the bit position is specified by `d3`.
	`d2` (lower byte of address)	`01` (`ACK`)	If any of the data at the position specified by `d3` (1) is '`1`', '`1`' will be stored in `P12`.
	`d3` (bit position)	`01` (`ACK`)	In all other cases, '`0`' will be stored in `P12`. Like command `80`, this command is a special command.
`81-FF`			Undefined.

11.6 Sample listings: send slave command

```
   0 :                     ; SLAVE
   0 :                     ; Send command to slave MCU
   0 :                     ; Send melody pattern to slave MCU and send command to play melody
   0 :                     ;
   0 :                     ; By K.A.
   0 :                     ;
   0 :                            PAGE   0
   0 :                            CPU    6301
1000 :                            ORG    $1000
1000 :                     ;
1000 :                     ;
1000 : =$11                TRCSR  EQU    $11
1000 : =$12                SRDR   EQU    $12
1000 : =$13                STDR   EQU    $13
1000 :                     ;
1000 :                     ; Set slave MCU melody (Yankee Doodle)
1000 : CE 10 22            PLAY   LDX    #MELTBL       ; (X): address where melody data is stored
1003 : 86 34                      LDAA   #$34          ; Send data to slave MCU
1005 : 8D 0E                      BSR    SNDSLV        ; Command 34: set melody data
1007 : A6 00               SLV10  LDAA   0,X           ; Set data
1009 : 8D 0A                      BSR    SNDSLV
100B : 08                         INX
100C : 81 FF                      CMPA   #$FF          ; Last datum is #$FF
100E : 26 F7                      BNE    SLV10
1010 :                     ;
1010 :                     ; Play melody
1010 : 86 35                      LDAA   #$35
1012 : 8D 01                      BSR    SNDSLV
1014 :                     ;
1014 : 39                         RTS
1015 :                     ;
1015 :                     ;
```

```
1015 :                      ; Subroutine
1015 :                      ; Send command to slave MCU
1015 :                      ; Parameter
1015 :                      ; On entry
1015 :                      ;   (A): command
1015 :                      ; On exit
1015 :                      ;   (A): received code
1015 :                      ; Register preserve: (X), (B)
1015 :                      ;
1015 : 7B 20 11             SNDSLV TIM    #$20,TRCSR    ; Tx ready?
1018 : 27 FB                       BEQ    SNDSLV
101A : 97 13                       STAA   STDR          ; Send command
101C :                      ; Receive from slave MCU
101C : 7D 00 11             SNDS10 TST    TRCSR         ; Rx ready?
101F : 2A FB                       BPL    SNDS10
1021 : 39                          RTS
1022 :                      ;
1022 :                      ;
1022 :                      ; Melody table (Yankee Doodle)
1022 : 29 0A 29 0A 0F 0A   MELTBL FCB    41,10,41,10,15,10,16,10
1028 : 10 0A
102A : 29 0A 10 0A 0F 0A          FCB    41,10,16,10,15,10,11,10
1030 : 0B 0A
1032 : 29 0A 29 0A 0F 0A          FCB    41,10,41,10,15,10,16,10
1038 : 10 0A
103A : 29 14 0D 0A 0B 0A          FCB    41,20,13,10,11,10
1040 : FF                         FCB    $FF
1041 :                      ;
1041 :                            END
```

Chapter 12

Bar-code reader

12.1 General

A bar code is a code which uses combinations of bars of varying thicknesses, designed to be read by an optical wand, and provides an effective means as a consumer product information code in inventory control, etc. (the current BASIC version of the HX-20 does not support the input/output of bar codes).

This chapter describes the methods of inputting bar codes and printing them out using MX-80 series printers (these functions will become available only with the external BASIC).

12.2 Input/output ports related to the bar-code reader

Input/output ports related to the bar-code reader are shown in Table 12.1 below.

MCU	Port	Direction	Function
Master MCU	`P20`	Input	Bar-code input signals (1: mark (black); 0: space (white)).
Slave MCU	`P35`	Output	Bar-code reader power supply (0: on; 1: off).
	`P41`	Output	Always 0.

Table 12.1: Input/output ports related to the bar-code reader.

Figure 12.1: Bar codes

When bar codes are to be scanned with a bar-code reader, each bar (black) is input as binary "1" (mark) and a blank (white) between bars is input as binary "0" (space) to the `P20` of the master MCU. A code is input by measuring the time duration of the black and white elements of the code.

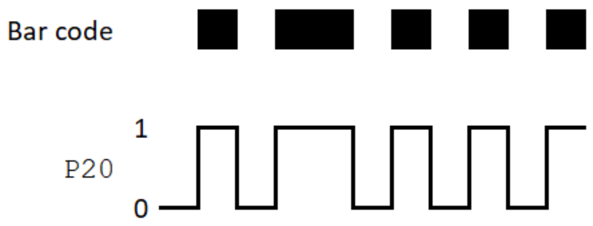


Figure 12.2: Bar code scanned and input signal

12.3 Procedure for data input

12.3.1 Turning on the power supply of the bar-code reader

Before inputting data to the bar-code reader, its power supply must be turned on as follows:

```
SNSCOM EQU   $FF19
       LDAA  #$03   ; Opens the special command mask of
                    ; the slave MCU
       JSR   SNSCOM
       LDAA  #$AA
       JSR   SNSCOM
       LDAA  #$08   ; Turns off the P35 of the slave MCU
       JSR   SNSCOM
       LDAA  #$00   ; Turns off the bit 5 at address 0006H
       JSR   SNSCOM
```

```
LDAA  #$6
JSR   SNSCOM
LDAA  #$DF
JSR   SNSCOM ; Special command of slave MCU
LDAA  #$04   ; Closes the special command masks
JSR   SNSCOM
```

12.3.2 Data input

Data must be input only after the power supply of the bar-code reader has been turned on. Data input is accomplished by measuring the time duration of the binary 1 or binary 0 at the port `P20` of the master MCU as follows:

Here, it is assumed that the initial state of `P20` is binary 1 (black).

1. Time measurement of the binary 1 (black) state

   (a) Set the bit 1 of `TCSR` to "0" (by specifying as a change of the input from "1" to "0"). ("`AIM #$FD,TCSR`").

   (b) Wait until the bit 7 of `TCSR` becomes "1" (indicating that the edge detection has been completed). The period of `FRC` at this point is approximately 0.1s. In normal bar-code scanning, the thickness of any single bar in a bar code will not exceed this time interval of 0.1s. Time-out monitoring is performed by the `OCR` in the bar-code reader so that any bar exceeding 0.1s in time duration may be detected as an error.

   Data setting in the `OCR` is performed as follows:

```
LDD   FRC    ; Sets the timing of the OCR
             ; interrupt at 0.1s
STD   OCR    ; 0.1s is judged
LDAB  TCSR   ; Clears the bit 6 (output compare
             ; flag) of TCSR
STAA  OCR
```

   Edge detection is performed as follows:

```
LOOP   LDAA TCSR   ; When bit7 = 1, it indicates that
                   ; edge detection is complete
       BMI  EDGE
       BITA #$40   ; Monitors the time-out condition
       BEQ  LOOP
       JMP  TIMOUT ; Executes the time-out processing
EDGE   LDD  ICR    ; (A,B): time duration of binary 1
       SUBD LSTTIM
       LDX  ICR    ; Stores the time when the edge is
       STX  LSTTIM ; detected
;      ...
LSTTIM RMB  2
```

2. Time measurement of the binary 0 (white) state

   The time duration of binary 0 is measured by the same procedure as described above, except the bit 1 of TCSR is set to “1” (by specifying as a change of the input edge from “0” to “1”).

12.3.3 Turning off the power supply of the bar-code reader

Upon completion of the data input to the bar-code reader, the power supply of the bar-code reader must be turned off as follows:

```
SNSCOM EQU   $FF19
       LDAA  #$03   ; Opens the special command mask of
                    ; the slave MCU
       JSR   SNSCOM
       LDAA  #$AA
       JSR   SNSCOM
       LDAA  #$07   ; Turns on the P35 of the slave MCU
       JSR   SNSCOM
       LDAA  #$00   ; Turns on the bit 5 at address 0006H
       JSR   SNSCOM
       LDAA  #$06
       JSR   SNSCOM
       LDAA  #$20
       JSR   SNSCOM
       LDAA  #$04   ; Closes the special command mask of
       JSR   SNSCOM ; the slave MCU
```

12.4 Printing bar codes with MX-80 series printers

The method of printing bar codes is explained using using the codes shown in Figure 12.3 as an example. The codes in this figure are available in two types of bars differing in thickness or width and two types of blanks differing in width. Namely, a 0.3mm narrow bar and a 1.0mm wide bar and a 0.3mm narrow blank and 1.0mm wide blank. To print these bars at a height of 1.7cm with any MX-80 series printer, the following must be specified:

1. Paper feed pitch: 4/216 inch (specified with `ESC, "3", 4`)
2. Dot density: 960 dots/line (specified with `ESC, "L`, n_1, n_2)

EPSON* EPSON* EPSON

Figure 12.3: Print sample of bar codes with MX-80

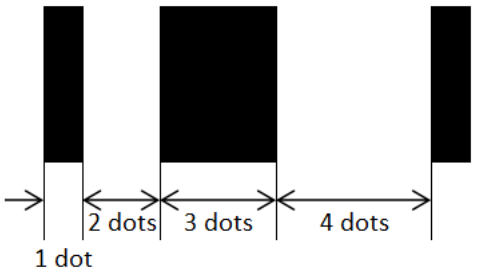


Figure 12.4: Print spacing with MX-80

The print spacing must be as shown in Figure 12.4.

- Narrow bar: 1-dot space.
- Wide bar: 3-dot space.
- Narrow blank: 2-dot space.
- Wide blank: 4-dot space.

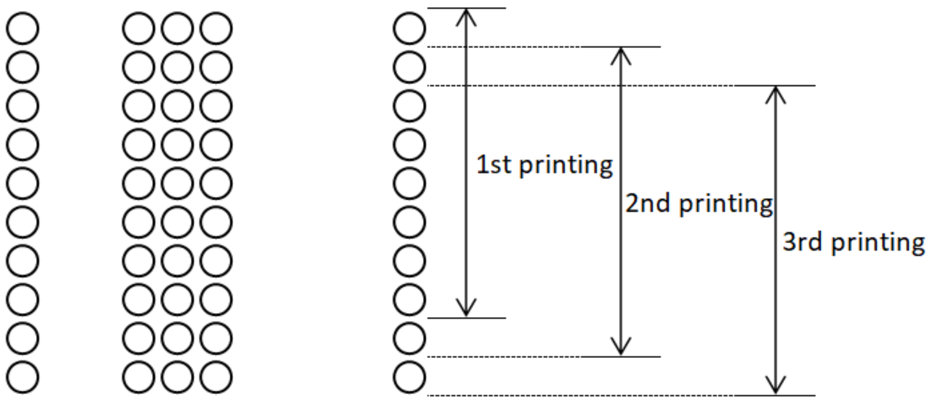


Figure 12.5: Repetition of graphic printing

A sequence of 8 dots is printed 16 times to produce a 1.7cm long bar (see Figure 12.5).

Bar codes listing:

0123456789

ABCDEFGHIJKLMNO

PQRSTUVWXYZ[$]^_

`abcdefghi

jklmno

pqrstuvwxy

z{|}^

'"#$%&()

,- :;<=>?@

12.5 Sample listings: reader decode program

```
   0 :                      ; BARCOD
   0 :                      ; Barcode reader read sample
   0 :                      ; Barcode reader decode program
   0 :                      ; Creative date: 1982/09/30
   0 :                      ;
   0 :                      ; By Koike
   0 :                           PAGE   0
   0 :                           CPU    6301
   0 :                      ;
   0 : =$3                  PORT2 EQU $3     ; Main PORT2 address
   0 : =$6                  PORT3  EQU    $6     ; Slave PORT3 address
   0 : =$7                  PORT4  EQU    $7     ; Slave PORT4 address
   0 : =$8                  TCSR   EQU    $8     ; Timer control status register
   0 : =$9                  FRC    EQU    $9     ; Free running counter
   0 : =$B                  OCR    EQU    $B     ; Output compare register
   0 : =$D                  ICR    EQU    $D     ; Input compare register
   0 : =$7C                 SIOSTS EQU    $7C    ; Slave I/O status
   0 : =$7D                 MIOSTS EQU    $7D    ; Main I/O status
   0 : =$76                 MINVAL EQU    118    ; Minimum width value
   0 : =$C350               OVRVAL EQU    50000  ; Overflow value
   0 :                      ;
   0 : =$FF16               RV232C EQU    $FF16  ; Slave RS232C recovery
   0 : =$FF19               SNSCOM EQU    $FF19  ; Slave communication
   0 :                      ;
   0 : =$5E2                HKLOAD EQU    $5E2   ; Hook load address for barcode
   0 : =$63C                HKABTD EQU    $63C   ; Hook abort address for barcode
   0 : =$665                DCBTAD EQU    $665   ; DCB pointer for barcode
   0 : =$68C                ASCFLG EQU    $68C   ; Load ASCII judge flag
   0 : =$68F                OPTNTB EQU    $68F   ; Option table address
   0 : =$8433               ERROR  EQU    $8433  ; BASIC error jump
   0 : =$8C70               FCERR  EQU    $8C70  ; BASIC FC-Error jump
   0 : =$A6D0               LODCNT EQU    $A6D0  ; BASIC continue loading address
   0 : =$A9D8            ABTDO  EQU    $A9D8  ; BASIC abort address
   0 :                   ;
   0 :                   ;
   0 :                   ; Test main
   0 :                   ;
1700 :                          ORG    $1700
1700 :                   ;
1700 : 86 01                    LDAA   #$01
1702 : B7 17 3D                 STAA   CHKDGT        ; Check digit judge flag
1705 : 86 01                    LDAA   #$01
1707 : B7 17 3E                 STAA   FULVER        ; Full ASCII judge flag
170A :                   ;
170A : BD 1B D3                 JSR    PONBAR        ; Power on barcode wand
170D :                   ;
170D : 86 02             MNST   LDAA   #$2           ; Gate open
170F : C6 07                    LDAB   #PORT4
1711 : BD 1C 0D                 JSR    SPWRIT
1714 :                   ;
1714 : 7F 17 FC                 CLR    ASCCNT
1717 : 7F 06 8C                 CLR    ASCFLG
171A :                   ;
171A : BD 18 01                 JSR    RECBAR        ; Recognition barcode
171D :                   ;
171D : B7 17 38                 STAA   ANSWER        ; Error code buffer
1720 : 25 06                    BCS    MN10
1722 : 7F 17 39                 CLR    CARRY         ; (C) buffer
1725 : 7E 17 30                 JMP    MN20
1728 :                   ;
1728 : 86 FF             MN10   LDAA   #$FF
172A : B7 17 39                 STAA   CARRY
172D : 7E 17 35                 JMP    MNED
1730 :                   ;
```

```
1730 : B6 17 38          MN20   LDAA   ANSWER
1733 : 27 D8                    BEQ    MNST
1735 :                   ;
1735 : 7E 17 35          MNED   JMP    *             ; Error end
1738 :                   ;
1738 :                   ANSWER RMB    1
1739 :                   CARRY  RMB    1
173A :                   ;
173A :                   ;
173A :                   ; Work area
173A :                   ;
173A :                   BAR    RMB    1      ; Bar bit pattern
173B :                   SPACE  RMB    1      ; Space bit pattern
173C :                   DIRECF RMB    1      ; Scan direction flag
173D :                   CHKDGT RMB    1      ; Check digit flag
173E :                   FULVER RMB    1      ; Full ASCII version judge flag
173F :                   CHRJDG RMB    1      ; Input routine first judge flag
1740 :                   TIMOVC RMB    2      ; Timer overflow counter
1742 :                   TIMCT1 RMB    2      ; Timer first counter
1744 :                   TIMCT2 RMB    2      ; Timer end counter
1746 :                   TIMCNT RMB    2      ; Timer counter
1748 :                   TIMSTC RMB    2      ;
174A :                   STRTMG RMB    2      ; Start margin
174C :                   SUMCHK RMB    2      ; Check digit sum
174E :                   ZNKBZC RMB    2      ; Last bar zero counter value
1750 :                   ZNKBOC RMB    2      ; Last bar one counter value
1752 :                   THRSHB RMB    2      ; Bar 1 or 0 threshold level
1754 :                   ZNKSZC RMB    2      ; Last space zero counter value
1756 :                   ZNKSOC RMB    2      ; Last space one counter value
1758 :                   THRSHS RMB    2      ; Space 1 or 0 threshold level
175A :                   ANSCTB RMB    1      ; Buffer
175B :                   FULHNT RMB    1      ; Full ASCII double character judge
```

```
175C :            ASCBF1 RMB    1      ; Double character first buffer
175D :            ASCBF2 RMB    1      ; Double character second buffer
175E :            BITCNT RMB    1      ; Character bit counter
175F :            ERRBF  RMB    1      ; Error code buffer
1760 :            ZNKOVF RMB    1      ; Zenkai overflow
1761 :            SPWRBF RMB    2      ; Slave write buffer
1763 :            ANSTBA RMB    2      ; Pre-answer table address
1765 :            ANSADR RMB    2      ;
1767 :            ANSASA RMB    2      ; Answer table address
1769 :            FULTBA RMB    2      ; Full ASCII table address
176B :            ;
176B :            ANSTBL RMB    72     ; Pre-answer table
17B3 :            ANSCNT RMB    1      ; Pre-answer counter
17B4 :            ;
17B4 :            ANSASC RMB    72     ; Answer table
17FC :            ASCCNT RMB    1      ; Answer counter
17FD :            ;
17FD : 00                FCB    0
17FE :            STDIGT RMB    1      ; Start check digit
17FF : 00                FCB    0
1800 :            EDDIGT RMB    1      ; End check digit
1801 :            ;
1801 :            ;
1801 :            ; Recognition
1801 :            ; Function: recognition barcode
1801 :            ; Call    : JSR RECBAR
1801 :            ; Return  : (A) = error code
1801 :            ;             000: normal
1801 :            ;             100: scan speed slower
1801 :            ;             101: scan speed faster
1801 :            ;           X 102: SW bad operation -- ver 0.3
1801 :            ;           X 103: timer overflow   -- ver 0.3
```

```
1801 :                  ;               104: buffer overflow
1801 :                  ;               105: not code39
1801 :                  ;               106: check digit error
1801 :                  ;               107: full ASCII conversion error
1801 :                  ;             (C) = break status
1801 :                  ;               0  : normal
1801 :                  ;               1  : break
1801 :                  ;
1801 :                  ;
1801 : 96 03            RECBAR LDAA   PORT2         ; Wand paper on?
1803 : 85 01                   BITA   #1
1805 : 27 13                   BEQ    REC50
1807 :                  ;
1807 : 7B B0 7D                TIM    #$B0,MIOSTS
180A : 27 F5                   BEQ    RECBAR
180C : 0D                      SEC
180D : 20 01                   BRA    REC800
180F :                  ;
180F :                  ;****** Return process ******
180F :                  ;
180F : 0C               REC700 CLC                  ; (C) clear
1810 :                  ;
1810 : 86 00            REC800 LDAA   #0            ; Answer count clear
1812 : B7 17 FC                STAA   ASCCNT
1815 :                  ;
1815 : 0E                      CLI                  ; Interrupt enable
1816 :                  ;
1816 : B6 17 5F                LDAA   ERRBF         ; Error code
1819 :                  ;
1819 : 39               REC900 RTS                  ; Return
181A :                  ;
181A :                  ;****** Margin detect ******
```

```
181A :                  ;
181A : 7B B0 7D         REC50  TIM    #$B0,MIOSTS
181D : 27 03                   BEQ    REC60
181F : 0D                      SEC
1820 : 20 EE                   BRA    REC800
1822 :                  ;
1822 : 96 08            REC60  LDAA   TCSR           ; ICF clear
1824 : DC 0D                   LDD    ICR
1826 :                  ;
1826 : DC 09                   LDD    FRC            ; OVF clear
1828 : FD 17 42                STD    TIMCT1
182B :                  ;
182B : C3 C3 50                ADDD   #OVRVAL        ; OCF clear
182E : DD 0B                   STD    OCR
1830 : 96 08                   LDAA   TCSR
1832 : 72 00 0B                OIM    #$0,OCR
1835 :                  ;
1835 : 72 02 08                OIM    #2,TCSR        ; IEDG=1
1838 :                  ;
1838 : BD 1A D8                JSR    TIMRED         ; Margin read
183B :                  ;
183B : 25 02                   BCS    REC70          ; Overflow?
183D : 20 1E                   BRA    REC90
183F :                  ;
183F : 96 08            REC70  LDAA   TCSR
1841 : 2B 02                   BMI    REC80          ; ICF?
1843 : 20 FA                   BRA    REC70
1845 :                  ;
1845 : DC 0D            REC80  LDD    ICR            ; Start value
1847 : FD 17 42                STD    TIMCT1
184A : C3 C3 50                ADDD   #OVRVAL        ; Overflow counter
184D : DD 0B                   STD    OCR
```

```
184F :                    ;
184F : 96 08                     LDAA   TCSR          ; OCF clear
1851 : 72 00 0B                  OIM    #$0,OCR
1854 : 71 FD 08                  AIM    #$FD,TCSR     ; IEDG=0
1857 :                    ;
1857 : CC FF FF           REC85  LDD    #$FFFF        ; Overflow counter set
185A : FD 17 46                  STD    TIMCNT
185D :                    ;
185D : FC 17 46           REC90  LDD    TIMCNT        ; Margin entry
1860 : FD 17 4A                  STD    STRTMG
1863 :                    ;
1863 : BD 1A D8                  JSR    TIMRED        ; Start bar read
1866 :                    ;
1866 : 24 02                     BCC    REC110
1868 :                    ;
1868 : 20 B0              REC100 BRA    REC50
186A :                    ;
186A : FC 17 46           REC110 LDD    TIMCNT
186D : 83 00 76                  SUBD   #MINVAL       ; 118 speed over?
1870 : 25 A8                     BCS    REC50
1872 :                    ;
1872 : FC 17 4A                  LDD    STRTMG        ; Margin check
1875 : 04                        LSRD                 ; (Margin) / 16 >= (Start bar width)
1876 : 04                        LSRD
1877 : 04                        LSRD
1878 : 04                        LSRD
1879 : B3 17 46                  SUBD   TIMCNT
187C : 25 9C                     BCS    REC50
187E :                    ;
187E :                    ;****** Start code ( * ) read ******
187E :                    ;
187E : FC 17 46                  LDD    TIMCNT        ; Initial value entry
```

```
1881 : FD 17 4E              STD    ZNKBZC        ; Initial narrow bar
1884 : FD 17 54              STD    ZNKSZC
1887 : 05                    ASLD
1888 : FD 17 50              STD    ZNKBOC        ; Initial wide value ( x2 )
188B : FD 17 56              STD    ZNKSOC        ;
188E :                ;
188E : F3 17 46              ADDD   TIMCNT        ; Threshold level entry ( x1.5 )
1891 : 04                    LSRD
1892 : FD 17 52              STD    THRSHB        ; Threshold level
1895 : FD 17 58              STD    THRSHS
1898 : 86 FF                 LDAA   #$FF          ; Pre-answer counter initial
189A : B7 17 B3              STAA   ANSCNT
189D :                ;
189D : 86 08                 LDAA   #8            ; Rest 8 bit of start code
189F : BD 1B 0B              JSR    DTTOBT
18A2 :                ;
18A2 : 27 02                 BEQ    REC120
18A4 : 20 C2                 BRA    REC100
18A6 :                ;
18A6 : B6 17 3A       REC120 LDAA   BAR           ; Normal direction check
18A9 : 81 06                 CMPA   #6
18AB : 26 0C                 BNE    REC130
18AD : B6 17 3B              LDAA   SPACE
18B0 : 81 08                 CMPA   #8
18B2 : 26 B4                 BNE    REC100        ; Not start code
18B4 : 7F 17 3C              CLR    DIRECF        ; L to R direction set
18B7 : 20 0E                 BRA    REC140
18B9 :                ;
18B9 : 81 0C          REC130 CMPA   #$C           ; Reverse direction check
18BB : 26 AB                 BNE    REC100        ; Not start code
18BD : B6 17 3B              LDAA   SPACE
18C0 : 81 01                 CMPA   #1
```

```
18C2 : 26 A4                   BNE    REC100
18C4 : B7 17 3C                STAA   DIRECF        ; R to L direction set
18C7 : 7B B0 7D         REC140 TIM    #$B0,MIOSTS   ; Break C
18CA : 27 04                   BEQ    REC150
18CC : 0D                      SEC
18CD : 7E 18 10                JMP    REC800
18D0 :                  ;
18D0 :                  ;****** Data read - in ******
18D0 :                  ;
18D0 : 0F               REC150 SEI                  ; Interrupt mask disable
18D1 :                  ;
18D1 : CC 00 00                LDD    #0
18D4 : FD 17 4C                STD    SUMCHK        ; Check digit sum area clear
18D7 : 86 FF                   LDAA   #$FF
18D9 : B7 17 FE                STAA   STDIGT        ; Start digit initia
18DC : CE 17 6B                LDX    #ANSTBL       ; Pre-answer table address
18DF : FF 17 63                STX    ANSTBA
18E2 :                  ;
18E2 : BD 1A D8         REC160 JSR    TIMRED        ; Character gap read
18E5 :                  ;
18E5 : 24 08                   BCC    REC170        ; Overflow?
18E7 : 86 64                   LDAA   #100          ; Scan speed slower
18E9 : B7 17 5F                STAA   ERRBF
18EC : 7E 18 0F                JMP    REC700
18EF :                  ;
18EF : 86 09            REC170 LDAA   #9            ; 1 character data bit convert
18F1 : BD 1B 0B                JSR    DTTOBT
18F4 :                  ;
18F4 : B7 17 5F                STAA   ERRBF
18F7 : 27 03                   BEQ    REC175
18F9 : 7E 18 0F                JMP    REC700
18FC :                  ;
```

```
18FC : B6 17 B3          REC175 LDAA   ANSCNT        ; Buffer over check
18FF : 81 49                    CMPA   #73
1901 : 25 08                    BCS    REC180
1903 : 86 68                    LDAA   #104          ; Buffer overflow error
1905 : B7 17 5F                 STAA   ERRBF
1908 : 7E 18 0F                 JMP    REC700
190B :                   ;
190B :                   ;****** Bit to ASCII code convert ******
190B :                   ;
190B : CE 1C 5E          REC180 LDX    #SPCTBL       ; Space table
190E : F6 17 3B                 LDAB   SPACE         ;(X) = (X) + (SPACE) x 4
1911 : 58                       ASLB
1912 : 58                       ASLB
1913 : 3A                       ABX
1914 :                   ;
1914 : B6 17 3C                 LDAA   DIRECF        ; Direction L to R?
1917 : 27 02                    BEQ    REC190
1919 : 08                       INX
191A : 08                       INX
191B :                   ;
191B : EC 00             REC190 LDD    0,X           ; Space table data
191D : 27 0B                    BEQ    REC200
191F : 2B 36                    BMI    REC280        ; Special character
1921 : 18                       XGDX
1922 : F6 17 3A                 LDAB   BAR
1925 : 3A                       ABX                  ; (X) = (X) + (BAR)
1926 : A6 00                    LDAA   0,X           ; Bar table data
1928 : 26 08                    BNE    REC210
192A :                   ;
192A : 86 69             REC200 LDAA   #105          ; Not code39 error
192C : B7 17 5F                 STAA   ERRBF
192F : 7E 18 0F                 JMP    REC700
```

```
1932 :                  ;
1932 :                  ;****** Check digit calculate ******
1932 :                  ;
1932 : 16               REC210 TAB                  ; (A) to (B)
1933 : C1 41                   CMPB   #$41         ; Alpha?
1935 : 24 1C                   BCC    REC250
1937 : C1 30                   CMPB   #$30         ; Numeric?
1939 : 24 14                   BCC    REC240
193B : C1 20                   CMPB   #$20         ; SP?
193D : 26 04                   BNE    REC220
193F : C6 26                   LDAB   #38          ; SP digit
1941 : 20 1B                   BRA    REC290
1943 :                  ;
1943 : C1 2D            REC220 CMPB   #$2D         ; - ?
1945 : 26 04                   BNE    REC230
1947 : C6 24                   LDAB   #36          ; - digit
1949 : 20 13                   BRA    REC290
194B :                  ;
194B : C6 25            REC230 LDAB   #37          ; . digit
194D : 20 0F                   BRA    REC290
194F :                  ;
194F : C0 30            REC240 SUBB   #$30         ; 0-9 digit
1951 : 20 0B                   BRA    REC290
1953 :                  ;
1953 : C0 37            REC250 SUBB   #$37         ; A-Z digit
1955 : 20 07                   BRA    REC290
1957 :                  ;
1957 : 84 7F            REC280 ANDA   #$7F         ; Special character
1959 : 7D 17 3A                TST    BAR
195C : 26 CC                   BNE    REC200       ; Not code39 error
195E :                  ;
195E : 81 2A            REC290 CMPA   #$2A         ; End code (*)?
```

```
1960 : 27 1E                   BEQ    REC310
1962 :                  ;
1962 : FE 17 63                LDX    ANSTBA
1965 : A7 00                   STAA   0,X           ; Answer ASCII entry
1967 : 08                      INX
1968 : FF 17 63                STX    ANSTBA        ; Next address save
196B : FE 17 4C                LDX    SUMCHK        ; Check digit sum
196E : 3A                      ABX
196F : FF 17 4C                STX    SUMCHK
1972 :                  ;
1972 : B6 17 FE                LDAA   STDIGT        ; Start digit?
1975 : 2A 03                   BPL    REC300
1977 : F7 17 FE                STAB   STDIGT
197A :                  ;
197A : F7 18 00         REC300 STAB   EDDIGT        ; New digit entry
197D : 7E 18 E2                JMP    REC160
1980 :                  ;
1980 :                  ; ****** Data arrangement and conversion ******
1980 :                  ;
1980 : 7A 17 B3         REC310 DEC    ANSCNT        ; Last (*) bun counter decrement
1983 :                  ;
1983 : 0E                      CLI                  ; Interrupt enable
1984 :                  ;
1984 : B6 17 3D                LDAA   CHKDGT
1987 : 27 32                   BEQ    REC340
1989 :                  ;
1989 : 7A 17 B3                DEC    ANSCNT        ; Check digit bun counter decrement
198C : B6 17 3C                LDAA   DIRECF        ; Direction check
198F : 26 0B                   BNE    REC320
1991 :                  ;
1991 : FC 17 4C                LDD    SUMCHK        ; L to R direction
1994 : B3 17 FF                SUBD   EDDIGT-1      ; Check sum
```

```
1997 : FD 17 4C                STD    SUMCHK
199A : 20 0F                   BRA    REC330
199C :                  ;
199C : FC 17 4C         REC320 LDD    SUMCHK        ; R to L direction
199F : B3 17 FD                SUBD   STDIGT-1      ; Check sum
19A2 : FD 17 4C                STD    SUMCHK
19A5 : B6 17 FE                LDAA   STDIGT        ; Last digit shitei
19A8 : B7 18 00                STAA   EDDIGT
19AB :                  ;
19AB : BD 1B C1         REC330 JSR    DGTCAL        ; Check digit calculate
19AE :                  ;
19AE : B1 18 00                CMPA   EDDIGT        ; Digit OK?
19B1 : 27 08                   BEQ    REC340
19B3 :                  ;
19B3 : 86 6A                   LDAA   #106          ; Check digit error
19B5 : B7 17 5F                STAA   ERRBF
19B8 : 7E 18 0F                JMP    REC700
19BB :                  ;
19BB :                  ;****** Answer ASCII arrangement ******
19BB :                  ;
19BB : CE 17 6B         REC340 LDX    #ANSTBL       ; Pre-answer table
19BE : CC 17 B4                LDD    #ANSASC       ; Answer table
19C1 : FD 17 67                STD    ANSASA
19C4 :                  ;
19C4 : B6 17 3C                LDAA   DIRECF        ; Direction flag
19C7 : 27 0B                   BEQ    REC360
19C9 : F6 17 B3                LDAB   ANSCNT        ; R to L direction
19CC : B6 17 3D                LDAA   CHKDGT
19CF : 26 02                   BNE    REC350
19D1 : C0 01                   SUBB   #1            ; None check digit
19D3 :                  ;
19D3 : 3A               REC350 ABX                  ; Top data address
```

```
19D4 :                ;
19D4 : FF 17 63       REC360 STX    ANSTBA      ; Source address
19D7 : B6 17 B3              LDAA   ANSCNT
19DA : B7 17 5A              STAA   ANSCTB      ; Transfer counter
19DD :                ;
19DD : B6 17 5A       REC370 LDAA   ANSCTB
19E0 : 27 28                 BEQ    REC400
19E2 :                ;
19E2 : FE 17 63              LDX    ANSTBA      ; Source address
19E5 : A6 00                 LDAA   0,X
19E7 : FE 17 67              LDX    ANSASA      ; Destination address
19EA : A7 00                 STAA   0,X
19EC : 08                    INX
19ED : FF 17 67              STX    ANSASA
19F0 :                ;
19F0 : B6 17 3C              LDAA   DIRECF
19F3 : 26 09                 BNE    REC380
19F5 : FE 17 63              LDX    ANSTBA      ; L to R direction
19F8 : 08                    INX                ; Next address
19F9 : FF 17 63              STX    ANSTBA
19FC : 20 07                 BRA    REC390
19FE :                ;
19FE : FE 17 63       REC380 LDX    ANSTBA      ; R to L direction
1A01 : 09                    DEX                ; Next address
1A02 : FF 17 63              STX    ANSTBA
1A05 :                ;
1A05 : 7A 17 5A       REC390 DEC    ANSCTB
1A08 : 20 D3                 BRA    REC370
1A0A :                ;
1A0A : B6 17 3E       REC400 LDAA   FULVER
1A0D : 26 19                 BNE    REC500
1A0F :                ;
```

```
1A0F :                 ;****** End process ******
1A0F :                 ;
1A0F : B6 17 B3               LDAA   ANSCNT        ; Pre-answer counter
1A12 : B7 17 FC               STAA   ASCCNT        ; Answer counter
1A15 :                 ;
1A15 : BD 1B A9        REC410 JSR    BEEPOK        ; OK beep
1A18 :                 ;
1A18 : 24 06                  BCC    REC420
1A1A : 7F 17 5F               CLR    ERRBF         ; Break
1A1D : 7E 18 10               JMP    REC800
1A20 :                 ;
1A20 : 7F 17 3F        REC420 CLR    CHRJDG        ; Normal end
1A23 : 4F                     CLRA                 ; (A) clear
1A24 : 0C                     CLC                  ; (C) clear
1A25 : 7E 18 19               JMP    REC900
1A28 :                 ;
1A28 :                 ;****** Full ASCII check ******
1A28 :                 ;
1A28 : CE 17 B4        REC500 LDX    #ANSASC
1A2B : FF 17 65               STX    ANSADR        ; Source address
1A2E : FF 17 67               STX    ANSASA        ; Destination address
1A31 : 7F 17 5B               CLR    FULHNT
1A34 : 7F 17 FC               CLR    ASCCNT        ; Answer counter clear
1A37 :                 ;
1A37 : B6 17 B3        REC510 LDAA   ANSCNT
1A3A : 81 00                  CMPA   #0
1A3C : 2F D7                  BLE    REC410        ; End
1A3E :                 ;
1A3E : 7A 17 B3               DEC    ANSCNT        ; Source counter decrement
1A41 : B6 17 5B               LDAA   FULHNT        ; Double character judge
1A44 : 27 08                  BEQ    REC520
1A46 : B6 17 5D               LDAA   ASCBF2        ; Special code ($,+,/,%)
```

```
1A49 : B7 17 5C                 STAA   ASCBF1
1A4C : 20 0C                    BRA    REC530
1A4E :                   ;
1A4E : FE 17 65          REC520 LDX    ANSADR        ; Source address
1A51 : A6 00                    LDAA   0,X
1A53 : B7 17 5C                 STAA   ASCBF1
1A56 : 08                       INX
1A57 : FF 17 65                 STX    ANSADR
1A5A :                   ;
1A5A : 81 24             REC530 CMPA   #$24          ; $
1A5C : 26 08                    BNE    REC540
1A5E : CE 1D 9E                 LDX    #FULASC
1A61 : FF 17 69                 STX    FULTBA
1A64 : 20 27                    BRA    REC580
1A66 :                   ;
1A66 : 81 2F             REC540 CMPA   #$2F          ; /
1A68 : 26 08                    BNE    REC550
1A6A : CE 1D B8                 LDX    #FULASC+26
1A6D : FF 17 69                 STX    FULTBA
1A70 : 20 1B                    BRA    REC580
1A72 :                   ;
1A72 : 81 2B             REC550 CMPA   #$2B          ; +
1A74 : 26 08                    BNE    REC560
1A76 : CE 1D D2                 LDX    #FULASC+52
1A79 : FF 17 69                 STX    FULTBA
1A7C : 20 0F                    BRA    REC580
1A7E :                   ;
1A7E : 81 25             REC560 CMPA   #$25          ; %
1A80 : 26 05                    BNE    REC570
1A82 : 7F 17 5B                 CLR    FULHNT
1A85 : 20 3F                    BRA    REC610
1A87 :                   ;
```

```
1A87 : CE 1D EC        REC570 LDX    #FULASC+78    ; %
1A8A : FF 17 69               STX    FULTBA
1A8D :                 ;
1A8D : B6 17 B3        REC580 LDAA   ANSCNT        ; Source data end check
1A90 : 81 00                  CMPA   #0
1A92 : 2F 32                  BLE    REC610        ; End data entry
1A94 :                 ;
1A94 : FE 17 65               LDX    ANSADR        ; Next data pre-read
1A97 : A6 00                  LDAA   0,X
1A99 : B7 17 5D               STAA   ASCBF2
1A9C : 08                     INX
1A9D : FF 17 65               STX    ANSADR        ; Next source address
1AA0 :                 ;
1AA0 : 81 41                  CMPA   #$41          ; Alpha?
1AA2 : 25 1D                  BCS    REC600
1AA4 :                 ;
1AA4 : 80 41                  SUBA   #$41
1AA6 : 16                     TAB
1AA7 : 7F 17 5B               CLR    FULHNT
1AAA : FE 17 69               LDX    FULTBA
1AAD : 3A                     ABX                  ; Data address
1AAE :                 ;
1AAE : A6 00                  LDAA   0,X           ; Conversion data
1AB0 : 81 FF                  CMPA   #$FF
1AB2 : 26 08                  BNE    REC590
1AB4 :                 ;
1AB4 : 86 6B                  LDAA   #107          ; Full ASCII error
1AB6 : B7 17 5F               STAA   ERRBF
1AB9 : 7E 18 0F               JMP    REC700
1ABC :                 ;
1ABC : 7A 17 B3        REC590 DEC    ANSCNT        ; Source counter decrement
1ABF : 20 08                  BRA    REC620
```

```
1AC1 :                 ;
1AC1 : 86 01           REC600 LDAA   #1            ; Single data flag set
1AC3 : B7 17 5B               STAA   FULHNT
1AC6 :                 ;
1AC6 : B6 17 5C        REC610 LDAA   ASCBF1        ; Pre-read data
1AC9 :                 ;
1AC9 : FE 17 67        REC620 LDX    ANSASA        ; Full ASCII entry
1ACC : A7 00                  STAA   0,X
1ACE : 08                     INX                  ; Destination next address
1ACF : FF 17 67               STX    ANSASA
1AD2 : 7C 17 FC               INC    ASCCNT        ; Destination counter renew
1AD5 : 7E 1A 37               JMP    REC510
1AD8 :                 ;
1AD8 :                 ;
1AD8 :                 ; Subroutine
1AD8 :                 ; Function: bar or space width timer value read
1AD8 :                 ; Call    : JSR TIMRED
1AD8 :                 ; Return  : (C) = return status
1AD8 :                 ;             0: normal
1AD8 :                 ;             1: overflow
1AD8 :                 ;           TIMCNT = timer value
1AD8 :                 ;
1AD8 :                 ;
1AD8 : 96 08           TIMRED LDAA   TCSR          ; Timer control status register
1ADA : 85 40                  BITA   #$40          ; Overflow check
1ADC : 26 24                  BNE    TIM100
1ADE : 85 80                  BITA   #$80          ; ICF check
1AE0 : 27 F6                  BEQ    TIMRED
1AE2 :                 ;
1AE2 : DC 0D                  LDD    ICR           ; Timer read
1AE4 : FD 17 44               STD    TIMCT2
1AE7 : B3 17 42               SUBD   TIMCT1
```

```
1AEA : FD 17 46                STD    TIMCNT        ; Timer value entry
1AED :                  ;
1AED : FC 17 44                LDD    TIMCT2        ; Start value renew
1AF0 : FD 17 42                STD    TIMCT1
1AF3 :                  ;
1AF3 : C3 C3 50                ADDD   #OVRVAL       ; Overflow counter set
1AF6 : DD 0B                   STD    OCR
1AF8 : 96 08                   LDAA   TCSR
1AFA : 72 00 0B                OIM    #$00,OCR
1AFD : 75 02 08                EIM    #2,TCSR       ; Edge convert
1B00 : 0C                      CLC                  ; Carry clear
1B01 :                  ;
1B01 : 39               TIM900 RTS                  ; Return
1B02 :                  ;
1B02 : CC FF FF         TIM100 LDD    #$FFFF        ; Overflow
1B05 : FD 17 46                STD    TIMCNT
1B08 : 0D                      SEC                  ; (C) set
1B09 : 20 F6                   BRA    TIM900
1B0B :                  ;
1B0B :                  ;
1B0B :                  ; Function: bar data read and bit convert
1B0B :                  ; Call    : JSR DTTOBT
1B0B :                  ;           (A)   = bit number
1B0B :                  ; Return  : (A)   = return status
1B0B :                  ;               0: normal
1B0B :                  ;             100: scan speed slower
1B0B :                  ;             101: scan speed faster
1B0B :                  ;           BAR   = bar bit answer
1B0B :                  ;           SPACE = space bit answer
1B0B :                  ;
1B0B :                  ;
1B0B : B7 17 5E         DTTOBT STAA   BITCNT        ; Bit counter
```

```
1B0E : 7F 17 3A                 CLR    BAR
1B11 : 7F 17 3B                 CLR    SPACE
1B14 :                   ;
1B14 : B6 17 5E          DTT10  LDAA   BITCNT        ; Bit end check
1B17 : 26 05                    BNE    DTT20
1B19 :                   ;
1B19 : 7C 17 B3                 INC    ANSCNT        ; End entry character renew
1B1C : 4F                       CLRA                 ; Normal return
1B1D :                   ;
1B1D : 39                DTT900 RTS                  ; Return
1B1E :                   ;
1B1E : B6 17 5E          DTT20  LDAA   BITCNT        ; Bar, space check
1B21 : 85 01                    BITA   #$1
1B23 : 26 05                    BNE    DTT30
1B25 : 78 17 3B                 ASL    SPACE         ; When space
1B28 : 20 03                    BRA    DTT40
1B2A :                   ;
1B2A : 78 17 3A          DTT30  ASL    BAR           ; When bar
1B2D :                   ;
1B2D : BD 1A D8          DTT40  JSR    TIMRED        ; Width read
1B30 :                   ;
1B30 : 24 04                    BCC    DTT50
1B32 :                   ;
1B32 : 86 64                    LDAA   #100          ; Scan speed slower error
1B34 : 20 E7                    BRA    DTT900
1B36 :                   ;
1B36 : FC 17 46          DTT50  LDD    TIMCNT
1B39 : 83 00 76                 SUBD   #MINVAL       ; 118 speed over check
1B3C : 24 04                    BCC    DTT60
1B3E :                   ;
1B3E : 86 65                    LDAA   #101          ; Scan speed faster
1B40 : 20 DB                    BRA    DTT900
```

```
1B42 :                ;
1B42 : B6 17 5E       DTT60  LDAA   BITCNT
1B45 : 85 01                 BITA   #$1         ; Bar or space check
1B47 : 26 2E                 BNE    DTT80       ; Bar
1B49 :                ;
1B49 :                ;****** Space ******
1B49 :                ;
1B49 : FC 17 46              LDD    TIMCNT
1B4C : B3 17 58              SUBD   THRSHS      ; Compare with space threshold
1B4F : 22 0F                 BHI    DTT70
1B51 :                ;
1B51 : FC 17 46              LDD    TIMCNT      ; When space 0
1B54 : FD 17 54              STD    ZNKSZC      ; Last space zero counter entry
1B57 : F3 17 56              ADDD   ZNKSOC
1B5A : 04                    LSRD               ; / 2
1B5B : FD 17 58              STD    THRSHS      ; New space threshold
1B5E : 20 43                 BRA    DTT110
1B60 :                ;
1B60 : FC 17 46       DTT70  LDD    TIMCNT      ; When space 1
1B63 : FD 17 56              STD    ZNKSOC      ; Last space one counter entry
1B66 : F3 17 54              ADDD   ZNKSZC
1B69 : 04                    LSRD               ; / 2
1B6A : FD 17 58              STD    THRSHS      ; New space threshold
1B6D :                ;
1B6D : B6 17 3B              LDAA   SPACE       ; Space bit set
1B70 : 8A 01                 ORAA   #1
1B72 : B7 17 3B              STAA   SPACE
1B75 : 20 2C                 BRA    DTT110
1B77 :                ;
1B77 :                ;****** Bar ******
1B77 :                ;
1B77 : FC 17 46       DTT80  LDD    TIMCNT
```

```
1B7A : B3 17 52            SUBD   THRSHB        ; Compare with bar threshold
1B7D : 22 0F               BHI    DTT90
1B7F :              ;
1B7F : FC 17 46            LDD    TIMCNT        ; When bar 0
1B82 : FD 17 4E            STD    ZNKBZC        ; Last bar zero counter entry
1B85 : F3 17 50            ADDD   ZNKBOC
1B88 : 04                  LSRD                 ; / 2
1B89 : FD 17 52            STD    THRSHB        ; New bar threshold
1B8C : 20 15               BRA    DTT110
1B8E :              ;
1B8E : FC 17 46     DTT90  LDD    TIMCNT        ; When bar 1
1B91 : FD 17 50            STD    ZNKBOC        ; Last bar one counter entry
1B94 : F3 17 4E            ADDD   ZNKBZC
1B97 : 04                  LSRD                 ; / 2
1B98 : FD 17 52            STD    THRSHB        ; New bar threshold
1B9B :              ;
1B9B : B6 17 3A            LDAA   BAR           ; Bar bit set
1B9E : 8A 01               ORAA   #1
1BA0 : B7 17 3A            STAA   BAR
1BA3 :              ;
1BA3 : 7A 17 5E     DTT110 DEC    BITCNT
1BA6 : 7E 1B 14            JMP    DTT10
1BA9 :              ;
1BA9 :              ;
1BA9 :              ; Function: OK beep on
1BA9 :              ; Call    : JSR BEEPOK
1BA9 :              ; Return  : (C) = break status
1BA9 :              ;             0: normal
1BA9 :              ;             1: break
1BA9 :              ;
1BA9 :              ;
1BA9 : BD 1C 53     BEEPOK JSR    SRWINT        ; Slave supervisor mask open
```

```
1BAC :              ;
1BAC : 86 30               LDAA   #$30          ; Slave beep command
1BAE : BD FF 19            JSR    SNSCOM
1BB1 :              ;
1BB1 : 86 1C               LDAA   #$1C          ; Sound level
1BB3 : BD FF 19            JSR    SNSCOM
1BB6 :              ;
1BB6 : 86 01               LDAA   #$1           ; Sound length
1BB8 : BD FF 19            JSR    SNSCOM
1BBB :              ;
1BBB : 25 03               BCS    BEP900
1BBD : BD FF 16            JSR    RV232C        ; Slave communication initial
1BC0 :              ;
1BC0 : 39           BEP900 RTS
1BC1 :              ;
1BC1 :              ;
1BC1 :              ; Function: chack digit calculate
1BC1 :              ; Call    : JSR DGTCAL
1BC1 :              ;           SUMCHK,+1 = check digit sum area
1BC1 :              ; Return  : (A)       = check digit
1BC1 :              ;
1BC1 :              ;
1BC1 : FC 17 4C     DGTCAL LDD    SUMCHK        ; Sum check
1BC4 : 83 00 2B            SUBD   #43
1BC7 : 25 05               BCS    DGT10
1BC9 :              ;
1BC9 : FD 17 4C            STD    SUMCHK
1BCC : 20 F3               BRA    DGTCAL
1BCE :              ;
1BCE : C3 00 2B     DGT10  ADDD   #43
1BD1 : 17                  TBA                  ; Rest (B) to (A)
1BD2 : 39                  RTS
```

```
1BD3 :                ;
1BD3 :                ;
1BD3 :                ; I/O subroutine
1BD3 :                ; Function: barcode wand power on
1BD3 :                ; Call    : JSR PONBAR
1BD3 :                ; Return  : (C) = return status
1BD3 :                ;             0: normal
1BD3 :                ;             1: break
1BD3 :                ;
1BD3 :                ;
1BD3 : 86 20          PONBAR LDAA   #$20          ; Barcode wand power on
1BD5 : C6 06                 LDAB   #PORT3
1BD7 : BD 1C 0D              JSR    SPWRIT
1BDA :                ;
1BDA : 25 06                 BCS    PON900        ; Break check
1BDC : 72 40 7C              OIM    #$40,SIOSTS   ; Power on status
1BDF :                ;
1BDF : BD FF 16              JSR    RV232C        ; Slave RS232C recovery
1BE2 :                ;
1BE2 : 39             PON900 RTS                  ; Return
1BE3 :                ;
1BE3 :                ;
1BE3 :                ; Function: barcode wand power off
1BE3 :                ; Call    : JSR POFBAR
1BE3 :                ; Return  : (C) = return status
1BE3 :                ;             0: normal
1BE3 :                ;             1: break
1BE3 :                ;
1BE3 :                ;
1BE3 : 86 20          POFBAR LDAA   #$20          ; Barcode wand power off
1BE5 : C6 06                 LDAB   #PORT3
1BE7 : CA 80                 ORAB   #$80
```

```
1BE9 : BD 1C 0D                 JSR    SPWRIT
1BEC :                   ;
1BEC : 25 06                    BCS    POF900        ; Break check
1BEE : 71 BF 7C                 AIM    #$BF,SIOSTS   ; Power on status
1BF1 :                   ;
1BF1 : BD FF 16                 JSR    RV232C        ; Slave RS232C recovery
1BF4 :                   ;
1BF4 : 39                POF900 RTS                  ; Return
1BF5 :                   ;
1BF5 :                   ;
1BF5 :                   ; Function: slave port read
1BF5 :                   ; Call    : JSR SPREAD
1BF5 :                   ; Return  : (A) = read data
1BF5 :                   ;           (C) = return status
1BF5 :                   ;             0: normal
1BF5 :                   ;             1: break
1BF5 :                   ;
1BF5 :                   ;
1BF5 : BD 1C 53          SPREAD JSR    SRWINT        ; Slave communication initial
1BF8 : 25 12                    BCS    SPR900        ; Error
1BFA :                   ;
1BFA : 86 05                    LDAA   #5            ; Read command
1BFC : BD FF 19                 JSR    SNSCOM
1BFF :                   ;
1BFF : 4F                       CLRA                 ; Port address (H)
1C00 : BD FF 19                 JSR    SNSCOM
1C03 :                   ;
1C03 : 17                       TBA                  ; Port address (L)
1C04 : BD FF 19                 JSR    SNSCOM
1C07 : 25 03                    BCS    SPR900        ; Error
1C09 :                   ;
1C09 : BD FF 16                 JSR    RV232C        ; Slave RS232C recovery
```

```
1C0C :                  ;
1C0C : 39               SPR900 RTS                  ; Return
1C0D :                  ;
1C0D :                  ;
1C0D :                  ; Function: slave port data write
1C0D :                  ; Call    : JSR SPWRIT
1C0D :                  ;           (A) = output data
1C0D :                  ;           (B) = port address
1C0D :                  ; Return  : (C) = return status
1C0D :                  ;             0: normal
1C0D :                  ;             1: break
1C0D :                  ;
1C0D :                  ;
1C0D : FD 17 61         SPWRIT STD    SPWRBF        ; Data save
1C10 : C4 7F                   ANDB   #$7F          ; Port address
1C12 :                  ;
1C12 : BD 1B F5                JSR    SPREAD        ; Port status read
1C15 : 25 3B                   BCS    SPW900        ; Error
1C17 : 36                      PSHA                 ; Data save
1C18 :                  ;
1C18 : B6 17 62                LDAA   SPWRBF+1
1C1B : 2B 11                   BMI    SPWR10
1C1D :                  ;
1C1D : B6 17 61                LDAA   SPWRBF        ; Data reset
1C20 : 88 FF                   EORA   #$FF
1C22 : B7 17 61                STAA   SPWRBF        ; Data invert
1C25 :                  ;
1C25 : 32                      PULA
1C26 : B4 17 61                ANDA   SPWRBF        ; Out data
1C29 : B7 17 61                STAA   SPWRBF
1C2C : 20 07                   BRA    SPWR20
1C2E :                  ;
```

```
1C2E : 32                SPWR10 PULA                  ; Data set
1C2F : BA 17 61                 ORAA   SPWRBF         ; Out data
1C32 : B7 17 61                 STAA   SPWRBF
1C35 :                   ;
1C35 : BD 1C 53          SPWR20 JSR    SRWINT         ; Slave communication initial
1C38 : 25 18                    BCS    SPW900         ; Error
1C3A :                   ;
1C3A : 86 06                    LDAA   #6             ; Write command
1C3C : BD FF 19                 JSR    SNSCOM
1C3F :                   ;
1C3F : 4F                       CLRA                  ; Port address (H)
1C40 : BD FF 19                 JSR    SNSCOM
1C43 :                   ;
1C43 : 17                       TBA                   ; Port address (L)
1C44 : BD FF 19                 JSR    SNSCOM
1C47 :                   ;
1C47 : B6 17 61                 LDAA   SPWRBF         ; Data output
1C4A : BD FF 19                 JSR    SNSCOM
1C4D : 25 03                    BCS    SPW900         ; Error
1C4F :                   ;
1C4F : BD FF 16                 JSR    RV232C         ; Slave RS232C recovery
1C52 :                   ;
1C52 : 39                SPW900 RTS                   ; Return
1C53 :                   ;
1C53 :                   ;
1C53 :                   ; Function: Slave communication initial
1C53 :                   ; Call    : JSR SRWINT
1C53 :                   ; Return  : (C) = return status
1C53 :                   ;             0: normal
1C53 :                   ;             1: break
1C53 :                   ;
1C53 :                   ;
```

```
1C53 : 86 03              SRWINT LDAA   #3             ; Slave supervisor mask open
1C55 : BD FF 19                  JSR    SNSCOM
1C58 :                    ;
1C58 : 86 AA                     LDAA   #$AA
1C5A : BD FF 19                  JSR    SNSCOM
1C5D :                    ;
1C5D : 39                        RTS                   ; Return
1C5E :                    ;
1C5E :                    ;***************
1C5E :                    ;* Space table *
1C5E :                    ;***************
1C5E :                    ;
1C5E : 00 00 00 00        SPCTBL FDB    0,0            ; 0000 error
1C62 :                    ;
1C62 : 1C 9E                     FDB    BARTBL         ; 0001 L to R
1C64 : 1D 7E                     FDB    BARTBL+$E0     ; 1000 R to L
1C66 :                    ;
1C66 : 1C DE                     FDB    BARTBL+$40     ; 0010 L to R
1C68 : 1D 3E                     FDB    BARTBL+$A0     ; 0100 R to L
1C6A :                    ;
1C6A : 00 00 00 00               FDB    0,0            ; 0011 error
1C6E :                    ;
1C6E : 1D 1E                     FDB    BARTBL+$80     ; 0100 L to R
1C70 : 1C FE                     FDB    BARTBL+$60     ; 0010 R to L
1C72 :                    ;
1C72 : 00 00 00 00               FDB    0,0            ; 0101 error
1C76 :                    ;
1C76 : 00 00 00 00               FDB    0,0            ; 0110 error
1C7A :                    ;
1C7A : A5                        FCB    $A5            ; 0111 L to R % code
1C7B : 2A                        FCB    $2A            ;             % digit
1C7C : A4                        FCB    $A4            ;      R to L $ code
```

```
1C7D : 27                          FCB    $27            ;              $ digit
1C7E :                       ;
1C7E : 1D 5E                       FDB    BARTBL+$C0     ; 1000 L to R
1C80 : 1C BE                       FDB    BARTBL+$20     ; 0001 R to L
1C82 :                       ;
1C82 : 00 00 00 00                 FDB    0,0            ; 1001 error
1C86 :                       ;
1C86 : 00 00 00 00                 FDB    0,0            ; 1010 error
1C8A :                       ;
1C8A : AB                          FCB    $AB            ; 1011 L to R + code
1C8B : 29                          FCB    $29            ;              + digit
1C8C : AF                          FCB    $AF            ;      R to L / code
1C8D : 28                          FCB    $28            ;              / digit
1C8E :                       ;
1C8E : 00 00 00 00                 FDB    0,0            ; 1100 error
1C92 :                       ;
1C92 : AF                          FCB    $AF            ; 1110 L to R / code
1C93 : 28                          FCB    $28            ;              / digit
1C94 : AB                          FCB    $AB            ;      R to L + code
1C95 : 29                          FCB    $29            ;              + digit
1C96 :                       ;
1C96 : A4                          FCB    $A4            ; 1110 L to R $ code
1C97 : 27                          FCB    $27            ;              $ digit
1C98 : A5                          FCB    $A5            ;      R to L % code
1C99 : 2A                          FCB    $2A            ;              % digit
1C9A :                       ;
1C9A : 00 00 00 00                 FDB    0,0            ; 1111 error
1C9E :                       ;
1C9E :                       ;*************
1C9E :                       ;* Bar table *
1C9E :                       ;*************
1C9E :                       ;
```

```
1C9E : 00 00 00            BARTBL FCB    0,0,0           ; Space=0001 L to R
1CA1 : 51                         FCB    $51             ; Q
1CA2 : 00                         FCB    0
1CA3 : 4E                         FCB    $4E             ; N
1CA4 : 54                         FCB    $54             ; T
1CA5 : 00 00                      FCB    0,0
1CA7 : 4C                         FCB    $4C             ; L
1CA8 : 53                         FCB    $53             ; S
1CA9 : 00                         FCB    0
1CAA : 50                         FCB    $50             ; P
1CAB : 00 00 00 00                FCB    0,0,0,0
1CAF : 4B                         FCB    $4B             ; K
1CB0 : 52                         FCB    $52             ; R
1CB1 : 00                         FCB    0
1CB2 : 4F                         FCB    $4F             ; O
1CB3 : 00 00 00                   FCB    0,0,0
1CB6 : 4D                         FCB    $4D             ; M
1CB7 : 00 00 00 00 00 00          FCB    0,0,0,0,0,0,0
1CBD : 00
1CBE : 00 00 00                   FCB    0,0,0           ; Space=0001 R to L
1CC1 : 4D                         FCB    $4D             ; M
1CC2 : 00                         FCB    0
1CC3 : 4F                         FCB    $4F             ; O
1CC4 : 50                         FCB    $50             ; P
1CC5 : 00 00                      FCB    0,0
1CC7 : 52                         FCB    $52             ; R
1CC8 : 53                         FCB    $53             ; S
1CC9 : 00                         FCB    0
1CCA : 54                         FCB    $54             ; T
1CCB : 00 00 00 00                FCB    0,0,0,0
1CCF : 4B                         FCB    $4B             ; K
1CD0 : 4C                         FCB    $4C             ; L
```

```
960/    1CD1 : 00                     FCB    0
961/    1CD2 : 4E                     FCB    $4E          ; N
962/    1CD3 : 00 00 00               FCB    0,0,0
963/    1CD6 : 51                     FCB    $51          ; Q
964/    1CD7 : 00 00 00 00 00 00      FCB    0,0,0,0,0,0,0
        1CDD : 00
965/    1CDE : 00 00 00               FCB    0,0,0        ; Space=0010 L to R
966/    1CE1 : 47                     FCB    $47          ; G
967/    1CE2 : 00                     FCB    0
968/    1CE3 : 44                     FCB    $44          ; D
969/    1CE4 : 4A                     FCB    $4A          ; J
970/    1CE5 : 00 00                  FCB    0,0
971/    1CE7 : 42                     FCB    $42          ; B
972/    1CE8 : 49                     FCB    $49          ; I
973/    1CE9 : 00                     FCB    0
974/    1CEA : 46                     FCB    $46          ; F
975/    1CEB : 00 00 00 00            FCB    0,0,0,0
976/    1CEF : 41                     FCB    $41          ; A
977/    1CF0 : 48                     FCB    $48          ; H
978/    1CF1 : 00                     FCB    0
979/    1CF2 : 45                     FCB    $45          ; E
980/    1CF3 : 00 00 00               FCB    0,0,0
981/    1CF6 : 43                     FCB    $43          ; C
982/    1CF7 : 00 00 00 00 00 00      FCB    0,0,0,0,0,0,0
        1CFD : 00
983/    1CFE : 00 00 00               FCB    0,0,0        ; Space=0010 R to L
984/    1D01 : 43                     FCB    $43          ; C
985/    1D02 : 00                     FCB    0
986/    1D03 : 45                     FCB    $45          ; E
987/    1D04 : 46                     FCB    $46          ; F
988/    1D05 : 00 00                  FCB    0,0
989/    1D07 : 48                     FCB    $48          ; H
```

```
 990/    1D08 : 49                      FCB    $49            ; I
 991/    1D09 : 00                      FCB    0
 992/    1D0A : 4A                      FCB    $4A            ; J
 993/    1D0B : 00 00 00 00             FCB    0,0,0,0
 994/    1D0F : 41                      FCB    $41            ; A
 995/    1D10 : 42                      FCB    $42            ; B
 996/    1D11 : 00                      FCB    0
 997/    1D12 : 44                      FCB    $44            ; D
 998/    1D13 : 00 00 00                FCB    0,0,0
 999/    1D16 : 47                      FCB    $47            ; G
1000/    1D17 : 00 00 00 00 00 00       FCB    0,0,0,0,0,0,0
         1D1D : 00
1001/    1D1E : 00 00 00                FCB    0,0,0          ; Space=0100 L to R
1002/    1D21 : 37                      FCB    $37            ; 7
1003/    1D22 : 00                      FCB    0
1004/    1D23 : 34                      FCB    $34            ; 4
1005/    1D24 : 30                      FCB    $30            ; 0
1006/    1D25 : 00 00                   FCB    0,0
1007/    1D27 : 32                      FCB    $32            ; 2
1008/    1D28 : 39                      FCB    $39            ; 9
1009/    1D29 : 00                      FCB    0
1010/    1D2A : 36                      FCB    $36            ; 6
1011/    1D2B : 00 00 00 00             FCB    0,0,0,0
1012/    1D2F : 31                      FCB    $31            ; 1
1013/    1D30 : 38                      FCB    $38            ; 8
1014/    1D31 : 00                      FCB    0
1015/    1D32 : 35                      FCB    $35            ; 5
1016/    1D33 : 00 00 00                FCB    0,0,0
1017/    1D36 : 33                      FCB    $33            ; 3
1018/    1D37 : 00 00 00 00 00 00       FCB    0,0,0,0,0,0,0
         1D3D : 00
1019/    1D3E : 00 00 00                FCB    0,0,0          ; Space=0100 R to L
```

```
1D41 : 33                         FCB    $33            ; 3
1D42 : 00                         FCB    0
1D43 : 35                         FCB    $35            ; 5
1D44 : 36                         FCB    $36            ; 6
1D45 : 00 00                      FCB    0,0
1D47 : 38                         FCB    $38            ; 8
1D48 : 39                         FCB    $39            ; 9
1D49 : 00                         FCB    0
1D4A : 30                         FCB    $30            ; 0
1D4B : 00 00 00 00                FCB    0,0,0,0
1D4F : 31                         FCB    $31            ; 1
1D50 : 32                         FCB    $32            ; 2
1D51 : 00                         FCB    0
1D52 : 34                         FCB    $34            ; 4
1D53 : 00 00 00                   FCB    0,0,0
1D56 : 37                         FCB    $37            ; 7
1D57 : 00 00 00 00 00 00          FCB    0,0,0,0,0,0,0
1D5D : 00
1D5E : 00 00 00                   FCB    0,0,0          ; Space=1000 L to R
1D61 : 2D                         FCB    $2D            ; -
1D62 : 00                         FCB    0
1D63 : 58                         FCB    $58            ; X
1D64 : 2A                         FCB    $2A            ; *
1D65 : 00 00                      FCB    0,0
1D67 : 56                         FCB    $56            ; V
1D68 : 20                         FCB    $20            ; Sp
1D69 : 00                         FCB    0
1D6A : 5A                         FCB    $5A            ; Z
1D6B : 00 00 00 00                FCB    0,0,0,0
1D6F : 55                         FCB    $55            ; U
1D70 : 2E                         FCB    $2E            ; .
1D71 : 00                         FCB    0
```

```
1D72 : 59                          FCB    $59          ; Y
1D73 : 00 00 00                    FCB    0,0,0
1D76 : 57                          FCB    $57          ; W
1D77 : 00 00 00 00 00 00           FCB    0,0,0,0,0,0,0
1D7D : 00
1D7E : 00 00 00                    FCB    0,0,0        ; Space=1000 R to L
1D81 : 57                          FCB    $57          ; W
1D82 : 00                          FCB    0
1D83 : 59                          FCB    $59          ; Y
1D84 : 5A                          FCB    $5A          ; Z
1D85 : 00 00                       FCB    0,0
1D87 : 2E                          FCB    $2E          ; .
1D88 : 20                          FCB    $20          ; Sp
1D89 : 00                          FCB    0
1D8A : 2A                          FCB    $2A          ; *
1D8B : 00 00 00 00                 FCB    0,0,0,0
1D8F : 55                          FCB    $55          ; U
1D90 : 56                          FCB    $56          ; V
1D91 : 00                          FCB    0
1D92 : 58                          FCB    $58          ; X
1D93 : 00 00 00                    FCB    0,0,0
1D96 : 2D                          FCB    $2D          ; -
1D97 : 00 00 00 00 00 00           FCB    0,0,0,0,0,0,0
1D9D : 00
1D9E :                      ;
1D9E :                      ;*******************************
1D9E :                      ;* Full ASCII conversion table *
1D9E :                      ;*******************************
1D9E :                      ;
1D9E : 01                   FULASC FCB    $01    ; $A= SOH
1D9F : 02                          FCB    $02    ; $B= STX
1DA0 : 03                          FCB    $03    ; $C= ETX
```

```
1081/    1DA1 : 04                    FCB    $04     ; $D= EOT
1082/    1DA2 : 05                    FCB    $05     ; $E= ENQ
1083/    1DA3 : 06                    FCB    $06     ; $F= ACK
1084/    1DA4 : 07                    FCB    $07     ; $G= BEL
1085/    1DA5 : 08                    FCB    $08     ; $H= BS
1086/    1DA6 : 09                    FCB    $09     ; $I= HT
1087/    1DA7 : 0A                    FCB    $0A     ; $J= LF
1088/    1DA8 : 0B                    FCB    $0B     ; $K= VT
1089/    1DA9 : 0C                    FCB    $0C     ; $L= FF
1090/    1DAA : 0D                    FCB    $0D     ; $M= CR
1091/    1DAB : 0E                    FCB    $0E     ; $N= SO
1092/    1DAC : 0F                    FCB    $0F     ; $O= SI
1093/    1DAD : 10                    FCB    $10     ; $P= DLE
1094/    1DAE : 11                    FCB    $11     ; $Q= DC1
1095/    1DAF : 12                    FCB    $12     ; $R= DC2
1096/    1DB0 : 13                    FCB    $13     ; $S= DC3
1097/    1DB1 : 14                    FCB    $14     ; $T= DC4
1098/    1DB2 : 15                    FCB    $15     ; $U= NAK
1099/    1DB3 : 16                    FCB    $16     ; $V= SYN
1100/    1DB4 : 17                    FCB    $17     ; $W= ETB
1101/    1DB5 : 18                    FCB    $18     ; $X= CAN
1102/    1DB6 : 19                    FCB    $19     ; $Y= EM
1103/    1DB7 : 1A                    FCB    $1A     ; $Z= SUB
1104/    1DB8 :                 ;
1105/    1DB8 : 21                    FCB    $21     ; /A= !
1106/    1DB9 : 22                    FCB    $22     ; /B= "
1107/    1DBA : 23                    FCB    $23     ; /C= #
1108/    1DBB : 24                    FCB    $24     ; /D= $
1109/    1DBC : 25                    FCB    $25     ; /E= %
1110/    1DBD : 26                    FCB    $26     ; /F= &
1111/    1DBE : 27                    FCB    $27     ; /G= ‘
1112/    1DBF : 28                    FCB    $28     ; /H= (
1DC0 : 29                       FCB    $29     ; /I= )
1DC1 : 2A                       FCB    $2A     ; /J= *
1DC2 : 2B                       FCB    $2B     ; /K= +
1DC3 : 2C                       FCB    $2C     ; /L= ´
1DC4 : FF                       FCB    $FF     ; /M= error
1DC5 : FF                       FCB    $FF     ; /N= error
1DC6 : 2F                       FCB    $2F     ; /O= /
1DC7 : 30                       FCB    $30     ; /P= 0
1DC8 : 31                       FCB    $31     ; /Q= 1
1DC9 : 32                       FCB    $32     ; /R= 2
1DCA : 33                       FCB    $33     ; /S= 3
1DCB : 34                       FCB    $34     ; /T= 4
1DCC : 35                       FCB    $35     ; /U= 5
1DCD : 36                       FCB    $36     ; /V= 6
1DCE : 37                       FCB    $37     ; /W= 7
1DCF : 38                       FCB    $38     ; /X= 8
1DD0 : 39                       FCB    $39     ; /Y= 9
1DD1 : 3A                       FCB    $3A     ; /Z= :
1DD2 :                  ;
1DD2 : 61                       FCB    $61     ; +A= a
1DD3 : 62                       FCB    $62     ; +B= b
1DD4 : 63                       FCB    $63     ; +C= c
1DD5 : 64                       FCB    $64     ; +D= d
1DD6 : 65                       FCB    $65     ; +E= e
1DD7 : 66                       FCB    $66     ; +F= f
1DD8 : 67                       FCB    $67     ; +G= g
1DD9 : 68                       FCB    $68     ; +H= h
1DDA : 69                       FCB    $69     ; +I= i
1DDB : 6A                       FCB    $6A     ; +J= j
1DDC : 6B                       FCB    $6B     ; +K= k
1DDD : 6C                       FCB    $6C     ; +L= l
1DDE : 6D                       FCB    $6D     ; +M= m
```

```
1145/    1DDF : 6E                          FCB     $6E     ; +N= n
1146/    1DE0 : 6F                          FCB     $6F     ; +O= o
1147/    1DE1 : 70                          FCB     $70     ; +P= p
1148/    1DE2 : 71                          FCB     $71     ; +Q= q
1149/    1DE3 : 72                          FCB     $72     ; +R= r
1150/    1DE4 : 73                          FCB     $73     ; +S= s
1151/    1DE5 : 74                          FCB     $74     ; +T= t
1152/    1DE6 : 75                          FCB     $75     ; +U= u
1153/    1DE7 : 76                          FCB     $76     ; +V= v
1154/    1DE8 : 77                          FCB     $77     ; +W= w
1155/    1DE9 : 78                          FCB     $78     ; +X= x
1156/    1DEA : 79                          FCB     $79     ; +Y= y
1157/    1DEB : 7A                          FCB     $7A     ; +Z= z
1158/    1DEC :                         ;
1159/    1DEC : 1B                          FCB     $1B     ; %A= ESC
1160/    1DED : 1C                          FCB     $1C     ; %B= FS
1161/    1DEE : 1D                          FCB     $1D     ; %C= GS
1162/    1DEF : 1E                          FCB     $1E     ; %D= RS
1163/    1DF0 : 1F                          FCB     $1F     ; %E= US
1164/    1DF1 : 3B                          FCB     $3B     ; %F= ;
1165/    1DF2 : 3C                          FCB     $3C     ; %G= <
1166/    1DF3 : 3D                          FCB     $3D     ; %H= =
1167/    1DF4 : 3E                          FCB     $3E     ; %I= >
1168/    1DF5 : 3F                          FCB     $3F     ; %J= ?
1169/    1DF6 : 5B                          FCB     $5B     ; %K= [
1170/    1DF7 : 5C                          FCB     $5C     ; %L= inv_slash
1171/    1DF8 : 5D                          FCB     $5D     ; %M= ]
1172/    1DF9 : 5E                          FCB     $5E     ; %N= ^
1173/    1DFA : 5F                          FCB     $5F     ; %O= _
1174/    1DFB : 7B                          FCB     $7B     ; %P= {
1175/    1DFC : 7C                          FCB     $7C     ; %Q= |
1176/    1DFD : 7D                          FCB     $7D     ; %R= }
```

```
1177/    1DFE : 7E                      FCB    $7E    ; %S= ~
1178/    1DFF : 7F                      FCB    $7F    ; %T= DEL
1179/    1E00 : 00                      FCB    $00    ; %U= NUL
1180/    1E01 : 40                      FCB    $40    ; %V= @
1181/    1E02 : 60                      FCB    $60    ; %W= '
1182/    1E03 : 7F                      FCB    $7F    ; %X= DEL
1183/    1E04 : 7F                      FCB    $7F    ; %Y= DEL
1184/    1E05 : 7F                      FCB    $7F    ; %Z= DEL
1185/    1E06 :                  ;
1186/    1E06 :                         END
```

Chapter 13

Miscellaneous I/O

13.1 Speaker output

Slave MCU port `15` supplies the output to the speaker. The required square wave frequencies are obtained by dividing this signal and outputting them to the piezoelectric speaker.

To obtain a 1000Hz output at the piezoelectric speaker, the output at port `15` should be as shown in Figure 13.1.

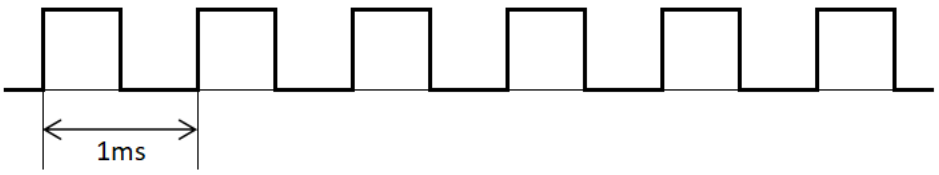


Figure 13.1: Output to the piezoelectric speaker

The `SOUND` subroutine has been provided to specify the tone and duration of the speaker output.

13.2 Expansion unit

The expansion unit features a 16KByte RAM and 16KByte ROM (only socket is provided). Addresses `0080` to `7FFF` can be used as RAM. A ROM, addresses `8000` to `BFFF`, may be selected by switching the HX-20 and expansion unit banks.

1. Memory area

When the expansion unit is connected, addresses `4000` to `7FFF` (16KBytes) may be used as a RAM. Data in the RAM is battery-backed up and protected. The ROM (`8000-BFFF`) is assigned as follows: bank 0 to the HX-20 and bank 1 to the expansion unit. Several memory configurations for the expansion unit are available. For details, refer to the hardware section of this manual.

2. Switching ROM banks

   When the ROM is mounted in the expansion unit, it is selected by switching banks. Banks are switched as follows:

   (a) To select the expansion unit ROM (bank 1), access address `0030` (either input or output is fine).

   (b) To select the ROM of the HX-20 (bank 0), access address `0032` or `0033` (either input or output is fine).

   None of these operations will be possible if the expansion unit is not connected to the HX-20. Also, switching can be performed for ROM area of the expansion unit. The HX-20 ROM (bank 0) is automatically selected when power is turned ON or upon reset.

13.3 Clock applications

The HX-20 clocks may be classified into two types: MCU clocks and IC clocks. The ports and registers related to clocks used in the HX-20 are as follows:

- MCU clocks
  - `OCR` (output compare register)
    1. Keyboard input sampling
       (uses `OCR` interrupt).
    2. RS-232C output timing setting
  - `ICR` (input capture register)
    1. Barcode reader timing setting
  - `TOF` (overflow of free running counter)
    1. Built-in microcassette counter sampling
- Real-time clock

  The real-time clock uses MCU area `4E` to `7F` as a RAM.

1. Use of clocks with application software

   (a) `OCR`

   An `OCF` interrupt is generated using `OCR` when a key on the keyboard is pressed. Sampling (key scanning) is then performed. Therefore, when `OCR` is used for this purpose, there is a strong chance that input from the keyboard will not be accepted.

   A function is also provided whereby, when `OCF` is set, RS-232C output will be performed by outputting the value of bit 0 of `TCSR` to `P21`.

   (b) `TOF`

   Counter sampling is executed using the `TOF` interrupt (at approx. 0.1s intervals) during I/O of files by the built-in microcassette.

   (c) `ICR`

   This register is used for barcode reader input. `ICR` measures the interval between pulse edges. However, barcode reader input software is not supported in the basic system of the HX-20.

   (d) Real-time clock

   The real-time clock is normally employed only to maintain the date and time. It can therefore be used freely in various applications. Sampling may be performed at intervals ranging from 4 to 500ms. Clock registers and RAMs are allocated as shown in Table 13.1.

Address	Input/Output	Description
0040	I/O	Seconds
0041	I/O	Alarm (seconds)
0042	I/O	Minutes
0043	I/O	Alarm (minutes)
0044	I/O	Hour
0045	I/O	Alarm (hour)
0046	I/O	Day
0047	I/O	Date
0048	I/O	Month
0049	I/O	Year
004A		Control register `A`
004B		Control register `B`

Continues in next page...

...continued from previous page

Address	Input/Output	Description
004C		Control register C
004D		Control register D
004E-007F		RAM 50 bytes

Table 13.1: Memory map of real-time clock

A 32.768Hz clock pulse is used as the master clock. RAM area 004E to 007F is used as an I/O flag area. Accessing this area can cause an I/O overrun.

13.4 Interrupts

MCU interrupt vectors are assigned as follows in ROM area FFEE-FFFF (Table 13.2).

Address	Value	Description
FFEE, FFEF	0106	TRAP
FFF0, FFF1	0109	SCI interrupt
FFF2, FFF3	010C	TOF interrupt
FFF4, FFF5	010F	OCF interrupt
FFF6, FFF7	0112	ICF interrupt
FFF8, FFF9	0115	IRQ1 (keyboard, power supply switch, clock, voltage down and external interrupts)
FFFA, FFFB	0118	SWI
FFFC, FFFD	011B	NMI
FFFE, FFFF	E000	Reset

Table 13.2: Interrupt vectors

Addresses 0106 to 011D are RAM addresses and addresses 0100 tio 0105 are used as entry points for interrupts. The initial values for addresses 0110 to 011D are stored in addresses FFB5 to FFCC (Table 13.3). Currently, 5 kinds of IRQ1 interrupts are supported.

Address	Description	Initialize timing
0100-0102	'JMP XXX' command. Referenced by IRQ1 interrupt routine (not supported in version 1) when IRQ1 clock interrupt is generated.	Reset (power ON)
0103-0105	'JMP XXX' command. Referenced by IRQ1 interrupt routine when external IRQ1 interrupt is generated.	Reset (power ON)
0106-0108	'JMP XXX' command (TRAP)	Reset (power ON)
0109-010B	'JMP XXX' command (SCI)	Reset (power ON)
010C-010E	'JMP XXX' command (TOF)	Reset (power ON)
010F-0111	'JMP XXX' command (OCF)	Reset (power ON)
0112-0114	'JMP XXX' command (ICF)	Reset (power ON)
0115-0117	'JMP XXX' command (IRQ1)	Reset (power ON)
0118-011A	Value not set (SW1)	No initialization
011B-011D	Value not set (NMI)	No initialization

Table 13.3: RAM area entry points for interrupt processing

Address	Description
FFB5-FFB7	'JMP XXX' command initial values for addresses 0100 to 0102 (entry point for clock interrupt)
FFB8-FFBA	'JMP XXX' command initial values for addresses 0103 to 0105
FFBB-FFBD	'JMP XXX' command initial values for addresses 0106 to 0108
FFBE-FFC0	'JMP XXX' command initial values for addresses 0109 to 010B
FFC1-FFC3	'JMP XXX' command initial values for addresses 010C to 010E
FFC4-FFC6	'JMP XXX' command initial values for addresses 010F to 0111
FFC7-FFC9	'JMP XXX' command initial values for addresses 0112 to 0114
FFCA-FFCC	'JMP XXX' command initial values for addresses 0115 to 0117

Table 13.4: Initial values for interrupt entry points

Item	Description	Interrupt confirmation	Interrupt mask
Keyboard	Interrupt is generated while a key is being pressed	`P15` = `0`	Set `P264` to ‘`0`’
Battery voltage	Interrupt is generated when the battery voltage falls below a specified level	`P14` = `0`	None
External interrupt	External bus terminal	`P13` = `0`	None
Power switch	Interrupt is generated when power switch is turned OFF	`P286` = `0`	None
Real-time clock	Real-time clock interrupt is generated	One of the address `004C` bits 4 to 7 is set to ‘`1`’	Set address `004B` bits 3 to 6 to ‘`0`’

Table 13.5: `IRQ1` interrupts

13.5 I/O initialization and termination

When the BREAK key is pressed, the interrupt processing routine issues a break command to the slave MCU to terminate the current I/O processing. Then bit 7 of address `007C` (variable name `SIOSTS`) and bit 7 of address `007D` (`MIOSTS`) are turned ON. When the bits 7 of `SIOSTS` and `MIOSTS` have been turned ON, the I/O routine assumes that I/O processing has been aborted by BREAK, sets the carry bit to logic ‘`1`’ and terminates processing.

The following subroutines have been provided to initialize or restart I/O processing:

1. I/O initialization

   Subroutine `INITIO` initializes I/O operations. Initialization is performed for the keyboard, LCD, microprinter, cassette I/O, ROM cartridge input and RS-232C input. Variables `SIOSTS` and `MIOSTS` are cleared. The serial communication driver is not informed. An initialize command is issued to the slave MCU.

2. I/O restart

Subroutine RSTRIO is used to restart I/O operations. Variables SIOSTS and MIOSTS are cleared. I/O flags for the external cassette, built-in microcassette, ROM cartridge and RS-232C port are also cleared. The microprinter output buffer is also cleared.

3. Warm start initialization

   Subroutine HSTRIO performs warm start initialization. The operation is identical to 1. I/O initialization above, except that keyboard and LCD initialization are not performed.

4. Cold start

   Subroutine REQINI is provided for cold start processing. The RAM is cleared when the current date and time are entered from the keyboard. The RAM area is checked and the last address of the RAM+1 and stored in addresses 012C, 012D and 0134, 0135. From this point, the processing is the same as that when power is turned ON.

13.6 Master MCU sleep

The master MCU may be set in sleep mode to reduce power consumption. The master MCU is reactivated when an interrupt is generated. In the current version, the master MCU enters the sleep mode while awaiting key input. There are restrictions on the sleep mode and subroutine SLEEP is called to set the master MCU in the sleep mode.

13.7 Output of address 26 port

The value of output port 26 is not actually read. Instead this value is set in address 004F (variable name 'P26') and the value of address 0026 can be obtained by inputting the contents of address 004F. Output of this value is performed by subroutine WRTP26.

13.8 General-purpose subroutines

Entry points have been provided for the following two general-purpose subroutines.

1. Subroutine HEXBIN converts an ASCII code hexadecimal number into a binary number.

2. Subroutine BINDEC converts an unsigned 16-bit binary number into an ASCII code decimal number.

13.9 Subroutine table

Subroutine name	Entry point	Description
`SOUND`	`FF64`	Sounds the speaker. • Parameters – At entry ∗ `(A)`: tone (`00`=pause; `06`=440Hz; `0D`=880Hz) `1`, `2`, `3`,...`1C`: 4-octave major scale (from C). `1D`, `1E`,...`38`: 4 octaves a half-tome higher than that for `1`, `2`, `3`,...`1C`. `39` to `FF`: assumed to be `0`. ∗ `(B)`: duration. `1` specified a duration of 0.1s. Duration may be specified in the range `01` to `FF`. Speaker is not activated when `00` is specified. – At return ∗ `(C)`: abnormal I/O flag. • Registers retained: `(A)`, `(B)`, `(X)`. • Subroutines referenced – `SNSCOM` – `SNSCOW` – `CHKRS` • Variables used: none

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
`SLEEP`	`FFA9`	Sets the master MCU in the sleep mode. Control is returned from the `SLEEP` subroutine when the sleep mode is exited. • Parameters – At entry: none – At return: none • Registers retained: `(A)`, `(B)`, `(C)`. • Subroutines referenced: none • Variables used: none

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents

`CHKPLG`	`FF64`	Identifies the plug-in options currently connected. The value of register `(A)` is also stored in variable `PLGSTS` (address `0079`).
		
		• Parameters
		  – At entry: none
		  – At return
		    ∗ `(C)`: abnormal I/O flag.
		    ∗ `(A)`: connected device code
		    Bit 2 Bit 1 Bit 0
		    `0` `0` `0` ROM cart.
		    `0` `0` `1` Reserved
		    `0` `1` `0` Unconnected
		    `0` `1` `1` Reserved
		    `1` x x Microcass.
		    (x: don't care)
		
		• Registers retained: `(B)`, `(X)`.
		
		• Subroutines referenced
		  – `SNSCOM`
		  – `CHKRS`
		
		• Variables used: none

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
`PWROFF`	`FFAC`	Turns power supply of the HX-20 OFF (the power switch is not actually turned OFF). There is therefore no exit from this subroutine. • Parameters – At entry: none – At return: none • Subroutines referenced: `SNSCOM` • Variables used: none

`PWRDWN`	`FF1F`	Displays the message "`CHARGE BATTERY!`" on the LCD. Control is returned from this subroutine when power supply voltage recovers. Otherwise, the power supply is turned OFF after the message has been flashed on the LCD 60 times.
		• Parameters
		  – At entry: none
		  – At return: none
		• Subroutines referenced
		  – `SNSCOM`
		  – `PWROFF`
		• Variables used: none

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
`REQINI`	`FF13`	Outputs the message "`Enter DATE and TIME`" at cold start. When the date and time are entered, the extent of the RAM is checked and the memory is cleared. Jumps to the entry point for reset. • Parameters – At entry: none – At return: none • Subroutines referenced – `DSPLCN` – `DSPLCH` – `KEYIN` – `HEXBIN`

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
`WRTP26`	`FED4`	Port `26` data output. This subroutine is used to output data to port `26`. Address `26` data is retained by address `4F`. • Parameters – At entry ∗ `(A)`: bit positions to be output (for each bit, ‘`1`’ indicates output and “`0`’, that the bit is not to be output). (To specify output of bits 0 and 1, set `03` in this register, that is, bits 0 and 1 ON). ∗ `(B)`: output data (bits not specified in `(A)` are ignored). – At return: none • Registers retained: `(A)`, `(B)`, `(X)`. • Subroutines referenced: none • Variables used: `R0H` (value is recovered).

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents

`BREKIO`	`FFA3`	I/O break. Executes break processing sequence for the slave MCU and turns bits 7 of variables `MIOSTS` and `SIOSTS` ON (logic '`1`').
		• Parameters
		  – At entry: none
		  – At return: none
		• Registers retained: none
		• Subroutines referenced
		  – `WRTP26`
		  – `SNSCOM`
		  – `RSONOF`
		• Variables used: none

`RSTRIO`	`FFA6`	Sets the value of variables to enable restarting of I/O processing after `BREAK`. Bits 0, 1 and 2 of the following variables are set to '`0`': `MIOSTS`, `SIOSTS`, `CSMOD` (external cassette status), `PRMSTS` (ROM cartridge status) and `SRSTS`. Print buffer is cleared and interrupt is enabled. • Parameters – At entry: none – At return: none • Registers retained: `(X)` • Subroutines referenced: none • Variables used: none

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
`CONTIO`	`FFAF`	Clears bits 7 of variables `MIOSTS` and `SIOSTS` and restarts RS-232C input. • Parameters – At entry: none – At return: none • Registers retained: none • Subroutines referenced – `CHKRS` • Variables used: none

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
`INITIO`	`FFCD`	Initializes I/O, keyboard and LCD. Sends command `02` to the slave MCU (initialize command). Subroutine `RSTRIO` initialize also performed. Identified plug-in options and removes interrupt mask. Does not perform initialization for serial communication. • Parameters – At entry: none – At return: none • Registers retained: none • Subroutines referenced – `INITKY` – `INITLC` – `SNSCOM` – `HSTRIO` – `RSTRIO` • Variables used: none

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
`HSTRIO`	`FED1`	Initializes I/O operation. Does not initialize keyboard and LCD. • Parameters – At entry: none – At return: none • Registers retained: none • Subroutines referenced – `SNSCOM` • Variables used: none

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
`HEXBIN`	`FF2B`	Converts an ASCII code hexadecimal number into a binary number. Data is not converted in series but only 1 byte of data can be converted. • Parameters – At entry: ∗ `(A,B)`: ASCII code 2-digit hexadecimal number – At return ∗ `(A)`: binary number (result of conversion). ∗ `(B)`: return code · `00`: normal · `01`: data error (`(A,B)` not in range `0` to `F`) ∗ `(Z)`: according to the value of `(B)`. • Registers retained: none • Subroutines referenced: none • Variables used: none • Other: reentrant

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
`BINDEC`	`FF28`	Converts unsigned 16-bit binary number into an ASCII code decimal number. • Parameters – At entry: ∗ `(A,B)`: unsigned 16-bit binary number. ∗ `(X)`: address for storing 5-byte result of conversion. Zeros are not suppressed. – At return: none • Registers retained: `(X)` • Subroutines referenced: none • Variables used: none • Other: reentrant

Continues in next page...

...continued from previous page.

Subroutine name	Entry point	Contents
`GETCLK`	`FF31`	Inputs the current date and time from the real-time clock (version 2 or better). • Parameters – At entry: ∗ `(X)`: starting address of the memory area where the input data is to be stored. Data is 6 bytes: month, day, year, hour, minutes, seconds. Each item is in a 2-digit BCD code (one byte). – At return: the result is entered in the specified memory address. • Registers retained: `(X)` • Subroutines referenced: none • Variables used: none

Continues in next page...

…continued from previous page.

Subroutine name	Entry point	Contents
SETCLK	FEF8	Sets the current date in the real-time clock (version 2 or better). • Parameters – At entry: ∗ `(X)`: the starting address of the memory area where the specified data is to be stored. The format of the data is the same as for `GETCLK`. – At return: none • Registers retained: none • Subroutines referenced: none • Variables used: none

13.10 Sample listings: alarm interrupt

```
      0 :                     ; ALARM
      0 :                     ; Alarm interrupt (BASIC)
      0 :                     ; Display current time.
      0 :                     ; The melody is played when minutes is updated (second = 00), because
      0 :                     ; alarm interrupt is caused and melody commands are sent to slave
      0 :                     ; MCU in interrupt routine
      0 :                     ;
      0 :                     ; By K.A.
      0 :                     ;
      0 :                     ; BASIC program
      0 :                     ; 10 CLS
      0 :                     ; 20 FOR I=&HB00 TO &HB06
      0 :                     ; 30 READ J
      0 :                     ; 40 POKE I,J
      0 :                     ; 50 NEXT I
      0 :                     ; 60 FOR I=&HB10 TO &HB45
      0 :                     ; 70 READ J
      0 :                     ; 80 POKE I,J
      0 :                     ; 90 NEXT I
      0 :                     ; 100 EXEC &HB00
      0 :                     ; 105 'Write interrupt vector
      0 :                     ; 110 POKE &H116,&H0B
      0 :                     ; 120 POKE &H117,&H10
      0 :                     ; 130 POKE &H7E,&H80
      0 :                     ; 135 'Enable alarm interrupt
      0 :                     ; 140 POKE &H4B,&H22
      0 :                     ; 150 POKE &H41,&H00
      0 :                     ; 160 POKE &H43,&HFF
      0 :                     ; 170 POKE &H45,&HFF
      0 :                     ; 180 LOCATE 5,2
      0 :                     ; 190 PRINT TIME$
      0 :                     ; 200 GOTO 180
```

```
  0 :                ; 1000 DATA &HFC,&HFF,&HCB,&HFD,&H0B,&H07,&H39
  0 :                ; 1010 DATA &H96,&H4C,&H2B,&H05,&HFE,&H0B,&H07,&H6E,&HD0
  0 :                ; 1020 DATA &HCE,&H0B,&H33,&H86,&H34,&HBD,&HFF,&H19,&HA6,&H00,&H36
  0 :                ; 1030 DATA &HBD,&HFF,&H19,&H32,&H08,&H81,&HFF,&H26,&HF4
  0 :                ; 1040 DATA &H86,&H35,&HBD,&HFF,&H19,&H3B
  0 :                ; 1050 DATA 17,06,44,06,17,06,44,06,17,06,14,06,16,06,15,06,13,18,&HFF
  0 :                ;
  0 :                ;
  0 :                       PAGE   0
  0 :                       CPU    6301
B00 :                       ORG    $B00
B00 :                ;
B00 :                ; Store interrupt vector
B00 : =$FF19         SNSCOM EQU    $FF19
B00 : =$FFCA         INTIR1 EQU    $FFCA  ; IRQ1 interrupt initial address
B00 :                ;
B00 : FC FF CB              LDD    INTIR1+1
B03 : FD 0B 07              STD    SAVADD
B06 : 39                    RTS
B07 :                ;
B07 :                SAVADD RMB    2
B09 :                ; IRQ1 interrupt routine
B10 :                       ORG    $B10
B10 : 96 4C                 LDAA   $4C           ; Is interrupt caused by clock?
B12 : 2B 05                 BMI    CLKINT
B14 : FE 0B 07              LDX    SAVADD
B17 : 6E 00                 JMP    0,X
B19 :                ;
B19 :                ; Send slave MCU data of melody
B19 : CE 0B 33       CLKINT LDX    #MELTBL       ; (X): address where melody data are stored
B1C : 86 34                 LDAA   #$34          ; Send data to slave MCU
B1E : BD FF 19              JSR    SNSCOM        ; Command 34: send melody data
```

```
B21 : A6 00              SLV10  LDAA   0,X            ; Set data
B23 : 36                        PSHA
B24 : BD FF 19                  JSR    SNSCOM
B27 : 32                        PULA
B28 : 08                        INX
B29 : 81 FF                     CMPA   #$FF           ; Last character is $FF
B2B : 26 F4                     BNE    SLV10
B2D :                    ;
B2D :                    ; Play melody
B2D : 86 35                     LDAA   #$35
B2F : BD FF 19                  JSR    SNSCOM
B32 : 3B                        RTI
B33 :                    ;
B33 :                    ; Melody table (For Elise)
B33 : 11 06 2C 06 11 06  MELTBL FCB    17,06,44,06,17,06,44,06
B39 : 2C 06
B3B : 11 06 0E 06 10 06         FCB    17,06,14,06,16,06,15,06
B41 : 0F 06
B43 : 0D 12                     FCB    13,18
B45 : FF                        FCB    $FF
B46 :                    ;
B46 :                           END
```

Chapter 14

Memory map

14.1 Memory allocation

The memory of HX-20 is divided into the following areas.

Address	Without expansion unit	With expansion unit	Applications
0000 to 004D	I/O ports		This area is used by I/O routines as work and flag area.
004E to 007F	RAM (real-time clock)		
0080 to 00FF	RAM		This area is used as a work area by the BASIC interpreter.
0100 to 04AF	RAM		This area is used by I/O routines as work area and I/O buffer.
04B0 to 0A3F	RAM		This area is used as a work area by the BASIC interpreter.
0A40 to 3FFF	RAM		
4000 to 5FFF	None	RAM (in expansion unit)	

Continues in next page...

...continued from previous page.

Address	Without expansion unit	With expansion unit	Applications
`6000` to `7FFF`	ROM (ROM5) (only socket provided)	RAM (in expansion unit)	
`8000` to `9FFF`	ROM (ROM4)	ROM (ROM2). Can be switched to ROM in expansion unit.	ROM in the HX-20 is the BASIC interpreter.
`A000` to `BFFF`	ROM (ROM3)	ROM (ROM1). Can be switched to ROM in expansion unit.	ROM in the HX-20 is the BASIC interpreter.
`C000` to `DFFF`	ROM (ROM2)		`C000` to `CFFF` is memory area for the BASIC interpreter. `D000` to `DFFF` contains Menu, Monitor and virtual screen routines.
`E000` to `FFFF`	ROM (ROM1)		This area is used by I/O routines.

Table 14.1: Memory map

14.2 Jump table

Jump tables show the entry points of various subroutines. Entry points are indicated by a 3-byte address specification. Initial byte specified `7E` (`JMP` command) followed by high and low bytes of the address.

Address (from)	(to)	Contents	Remarks	Details in Chapter:
FED1	FED3	JMP HSTRIO	I/O restart, initialize	13
FED4	FED6	JMP WRTP26	Address 26 port output	13
FED7	FED9	JMP BILOAD	Memory load: load, close after end of processing	9
FEDA	FEDC	JMP OPNLOD	Memory load: load open	9
FEDD	FEDF	JMP BIDUMP	Memory dump: dump and close after end of processing	9
FEE0	FEE2	JMP OPNLOD	Memory dump: dump open	9
FEE3	FEE5	JMP DIRPRM	Read PROM cartridge directory	8
FEE6	FEE8	JMP CLSPRM	Closes PROM cartridge file	8
FEE9	FEEB	JMP REDPRM	Reads 1 character from PROM cartridge file.	8
FEEC	FEEE	JMP OPNPRM	Opens PROM cartridge file.	8
FEEF	FEF1	JMP CNTMCS	Read/write to built-in microcassette counter value.	6
FEF2	FEF4	JMP SECMCS	Advances tape to the specified built-in microcassette counter.	6
FEF5	FEF7	JMP REWMCS	Rewinds built-in microcassette	6
FEF8	FEFA	JMP SETCLK	Inputs time and date (version 2 or better)	13
FEFB	FEFD	JMP CLSMCS	ACloses built-in microcassette files.	6
FEFE	FF00	JMP WRTMCS	Outputs one character to built-in microcassette.	6
FF01	FF03	JMP OPNWMS	Opens built-in microcassette file for output.	6
FF04	FF06	JMP READMS	Inputs one character from built-in microcassette.	6
FF07	FF09	JMP OPNRMS	Opens built-in microcassette file for input (initializes file).	6
FF0A	FF0C	JMP MCSMAN	Opens built-in microcassette file for input (searches specified file).	6

Continues in next page...

...continued from previous page.

Address (from)	(to)	Contents	Remarks	Details in Chapter:
FF0D	FF0F	JMP SECMCS	Sets built-in microcassette in manual operation mode.	6
FF10	FF12	JMP $DFF7	Jumps to address DFF7 (Monitor).	
FF13	FF15	JMP REQINI	Initializes HX-20 cold start.	13
FF16	FF18	JMP CHKRS	RS-232C recovery after aborting input processing.	5
FF19	FF1B	JMP SNSCOM	Sends one command byte to slave MCU.	11
FF1C	FF1E	JMP SRINIT	Initializes high-speed serial communication.	4
FF1F	FF21	JMP PWRDWN	Battery low message.	13
FF22	FF24	JMP KYSSTK	Stores data in the initial key stack.	2
FF25	FF27	JMP $DFFD	Jumps to address DFFD (MENU).	
FF28	FF2A	JMP BINDEC	Converts binary numbers into ASCII decimal code.	13
FF2B	FF2D	JMP HEXBIN	Converts ASCII hexadecimal code into binary code.	13
FF2E	FF30	JMP CHKPLG	CIdentification of plug-in options.	13
FF31	FF33	JMP GETCLK	Sets time and date (version 2 or better)	13
FF34	FF36	JMP CLSCS	Closes external cassette file.	6
FF37	FF39	JMP WRITCS	Outputs one byte to external cassette file.	6
FF3A	FF3C	JMP OPNWCS	Opens external cassette file for output.	6
FF3D	FF3F	JMP READCS	Inputs 1 byte from external cassette file.	6
FF40	FF42	JMP SRCRCS	Opens external cassette file for input (initializes file).	6
FF43	FF45	JMP OPNRCS	Opens external cassette file for input.	6

Continues in next page...

...continued from previous page.

Address (from)	(to)	Contents	Remarks	Details in Chapter:
FF46	FF48	JMP PONFCS	External cassette file remote (ON/OFF).	6
FF49	FF4B	JMP DSPLCN	Displays n characters on LCD (physical screen).	3
FF4C	FF4E	JMP DSPLCH	Displays one character on LCD (physical screen).	3
FF4F	FF51	JMP $DFF1	Displays one character on virtual screen.	15
FF52	FF54	JMP LCADDR	Link table for LCD routines. Selects LCD driver.	
FF55	FF57	JMP LCDMOD	Link table for LCD routines. Selects LCD driver mode.	
FF58	FF5A	JMP DATMOD	Link table for LCD routines. Outputs data to LCD driver.	
FF5B	FF5D	JMP DISPIT	Displays one character on LCD (data is not entered in physical screen buffer)	3
FF5E	FF60	JMP $DFF4	Calls virtual screen function	15
FF61	FF63	JMP $DFEE	Displays (recovers) current virtual screen data.	15
FF64	FF66	JMP SOUND	Speaker output.	13
FF67	FF69	JMP CHRGEN	Generates character font.	3
FF6A	FF6C	JMP KEYSCN	Scans key matrix.	2
FF6D	FF6F	JMP SERIN	High-speed serial data input.	4
FF70	FF72	JMP SEROUT	High-speed serial data output.	4
FF73	FF74	JMP SERONF	High-speed driver ON/OFF.	4
FF76	FF78	JMP RSPUT	Outputs one character to RS-232C.	5
FF79	FF7B	JMP RSGET	Imputs one character from RS-232C.	5
FF7C	FF7E	JMP RSGSTS	Inputs RS-232C status register value.	5
FF7F	FF81	JMP RSCLOS	Closes RS-232C input.	5
FF82	FF84	JMP RSOPEN	Opens RS-232C output.	5

Continues in next page...

...continued from previous page.

Address (from)	(to)	Contents	Remarks	Details in Chapter:
FF85	FF87	JMP RSONOF	Controls RS-232C driver (ON/OFF).	5
FF88	FF8A	JMP RSMST	Sets RS-232C status register mode.	5
FF8B	FF8D	JMP SCRCPY	Screen copy (LCD to micro-printer).	7
FF8E	FF90	JMP NFEED	Performs n dot-lines of line feed on microprinter.	
FF91	FF93	JMP PRTDOT	Prints one dot-line (bit pattern) on the microprinter.	7
FF94	FF96	JMP LNPRNT	Prints one character-line on the microprinter.	7
FF97	FF99	JMP CHPRNT	Prints one character on the microprinter.	7
FF9A	FF9C	JMP KEYIN	Enters one character from keyboard.	2
FF9D	FF9F	JMP KEYSTS	Enters keyboard key status.	2
FFA0	FFA2	JMP INITKY	Initializes keyboard.	2
FFA3	FFA5	JMP BREKIO	I/O break.	13
FFA6	FFA8	JMP RSTRIO	Restart after I/O break.	13
FFA9	FFAB	JMP SLEEP	Master MCU sleep.	13
FFAC	FFAE	JMP PWROFF	Power supply OFF.	13
FFAF	FFB1	JMP CONTIO	Continuation after I/O break.	13
FFB2	FFB4	JMP BRKIN	Entry point after BREAK key has been pressed.	
FFB5	FFB7	JMP CLKINT	Initial value for clock interrupt entry point.	
FFB8	FFBA	JMP IRQI80	Initial value for IRQ1 external interrupt entry point.	
FFBB	FFBD	JMP SDFFA	Initial value for TRAP interrupt entry point.	
FFBE	FFC0	JMP SERINT	Initial value for SCI interrupt entry point.	
FFC1	FFC3	JMP TOFINT	Initial value for TOF interrupt entry point.	

Continues in next page...

...continued from previous page.

Address (from)	(to)	Contents	Remarks	Details in Chapter:
FFC4	FFC6	JMP OCFINT	Initial value for OCF interrupt entry point.	
FFC7	FFC9	JMP ICFINT	Initial value for ICF interrupt entry point.	
FFCA	FFCC	JMP IRQINT	Initial value for IRQ1 interrupt entry point.	
FFCD	FFCF	JMP INITIO	I/O initialize.	

Table 14.2: Jump table ROM1 (E000 to FFFF)

Address (from)	(to)	Contents	Notes
DFEE	DFF0	JMP LCRECV	Covers the virtual screen and rewrites only the physical screen.
DFF1	DFF3	JMP SCRCHR	Displays one character on the virtual screen.
DFF4	DFF6	JMP SCRFNC	Screen functions of the virtual screen.
DFF7	DFF9	JMP MON	Monitor entry.
DFFA	DFFC	JMP MONTRP	Monitor entry on TRAP.
DFFD	DFFF	JMP MENU	Menu entry.

Table 14.3: Jump table ROM2 (C000 to DFFF)

14.3 ROM vectors

Address (from)	(to)	Variable name	Number of bytes	Description
FFD0	FFD1	NEWKTB	2	Shows the address at which the matrix data is stored after key scanning.
FFD2	FFD3	COLCNT	2	Shows the address where the amount of data in the built-in microprinter buffer is stored.
FFD4	FFD5	CSBFCM	2	Shows the address where the amount of data on the external cassette buffer is stored. Used for data read and write.
FFD6	FFD7	MSBFCM	2	Shows the address where the amount of data on the built-in microcassette buffer is stored. Used for data read and write.
FFD8	FFD9	RSDCNT	2	Shows the address where the amount of data on the RS-232C input buffer is stored.
FFDA	FFDB		2	Shows the starting address of the LCD physical screen buffer.
FFDC	FFDD	CASBUF	2	Shows the address of the 260-byte buffer used by the monitor.
FFDE	FFDF		2	Shows the address where the scroll speed data is stored.
FFE0	FFE1	CSHBUF	2	Shows the starting address of the external cassette header buffer.
FFE2	FFE3	MSHBUF	2	Shows the starting address of the built-in microcassette header buffer.
FFE4	FFE5	KEYMOD	2	Shows the address where the key input mode data is stored.
FFEE	FFEF		2	Shows the address where the TRAP entry point is stored. Set to 0106.
FFF0	FFF1		2	Shows the address where the SCI interrupt entry point is stored. Set to 0109.

Continues in next page...

…continued from previous page.				
Address (from)	(to)	Variable name	Number of bytes	Description
FFF2	FFF3		2	Shows the address where the TOF interrupt entry point is stored. Set to 010C.
FFF4	FFF5		2	Shows the address where the OCF interrupt entry point is stored. Set to 010F.
FFF6	FFF7		2	Shows the address where the ICF interrupt entry point is stored. Set to 0112.
FFF8	FFF9		2	Shows the address where the IRQ1 interrupt entry point is stored. Set to 0115.
FFFA	FFFA		2	Shows the address where the SWI interrupt entry point is stored. Set to 0118.
FFFC	FFFD		2	Shows the address where the NMI interrupt entry point is stored. Set to 011B.
FFFE	FFFF		2	Shows the address where the RESET interrupt entry point is stored. Set to E000.

Note: addresses are shown as two bytes in upper- and lower-byte sequence.

14.4 RAM page 0 vectors

Address (from)	Address (to)	Variable name	Number of bytes	Description
4E	4E	PWRFLG	1	• Bits 0 to 3: reserved for selecting processing to be executed when power supply is turned ON. • Bits 4 to 7: indicate the processing to be executed when power supply is turned OFF. Bit 7 6 5 4 0 0 0 0 No operation 0 0 0 1 Executes 0 0 1 0 the subroutine specified in addresses 132-133 (POFADR) prior to turning OFF the power supply. Other bit values No operation
4F	4F	P26	1	Address 26 port data. **Note:** read of address 26 is inhibited.
50	51	R0	2	This area is used as a work area by I/O routine.
52	53	R1	2	Same as R0.
54	55	R2	2	Same as R0.
56	57	R3	2	Same as R0.
58	59	R4	2	Same as R0.
5A	5B	R5	2	Same as R0.
5C	5D	R6	2	Same as R0.
5E	5F	R7	2	Same as R0.
60	61	M0	2	This area is used as a work area by Monitor and screen routines.
62	63	M1	2	Same as M0.
64	65	M2	2	Same as M0.
66	67	M3	2	Same as M0.

Continues in next page...

...continued from previous page.

Address (from)	(to)	Variable name	Number of bytes	Description
68	69	M4	2	Same as M0.
6A	6B	M5	2	Same as M0.
6C	6D	M6	2	Same as M0.
6E	6F	M7	2	Same as M0.
70	71	K0	2	This area is used as a work area by the key input routine.
72	73	K1	2	Same as K0.
74	75	S0	2	Same as K0.
76	77	S1	2	Same as K0.
78	78	INIFL1	1	Indicates application program cold start. For each bit, '0' indicates cold start and '1'. warm start. • Bits 0 to 5: • Bit 6: BASIC application programs. • Bit 7: BASIC interpreter.

Continues in next page...

...continued from previous page.

Address (from)	(to)	Variable name	Number of bytes	Description
79	79	PLGSTS	1	• Bits 0 to 2: indicate the plug-in option. Bit 2 / 1 / 0: 0 0 0 ROM cassette 0 0 1 Reserved 0 1 0 Not connected 0 1 1 Reserved 1 x x Microcassette (x: don't care) • Bit 3: 0. • Bits 4 to 6: not used. • Bit 7: specifies whether RS-232C driver will be turned OFF when the BREAK key is pressed – 0: not turned OFF. – 1: turned OFF.

Continues in next page...

...continued from previous page.				
Address (from)	Address (to)	Variable name	Number of bytes	Description
7A	7A	SRSTS	1	Bits 0 to 2: indicate current RS-232C status. b2 1 0 0 0 0 Input operation is not being performed. 0 0 1 Input operation is being executed. 0 1 x Not used in current version 1 0 0 Undefined 1 0 1 Input. Operation enters wait state when the slace MCU is busy with other I/O devices such as the microprinter. 1 1 x Undefined Bit 3: indicates RS-232C driver status (0: OFF; 1: ON). Bit 4: serial I/F driver status. The same driver is used as the RS-232C and serial I/F driver. However, in terms of operation by software, they are treated independently. Bits 5 to 7: SCI (serial communication interface) interrupt mode b7 6 5 0 0 0 Input of external cassette data. 0 0 1 Input of internal microcassette data. 0 1 0 RS-232C data input. 0 1 1 Serial I/F data input. 1 0 0 Output of external cassette data. 1 0 1 Output of internal microcassette data. 1 1 x Undefined

Continues in next page...

...continued from previous page.

Address (from)	Address (to)	Variable name	Number of bytes	Description
`7B`	`7B`	`RUNMOD`	1	Program execution mode • Bits 0 to 3: reserved for specifying program number, etc. • Bits 4 to 5: undefined. • Bit 6: flag indicating whether the virtual screen is being used – `0`: virtual screen being used. – `1`: virtual screen not being used. • Bit 7: indicates the interpreter mode – `0`: machine language mode. – `1`: interpreter mode. **Note:** in machine language mode, the program jumps to the specified address when the `BREAK` key is pressed, power is turned OFF or the voltage falls. In interpreter mode when one of these interrupts is generated, the appropriate flag is set (`MIOSIS`) and control is returned. In BASIC, the values of bits 7 and 6 are `1`, `0` and in Monitor `0`, `1`

Continues in next page...

...continued from previous page.

Address (from)	Address (to)	Variable name	Number of bytes	Description
`7C`	`7C`	`SIOSTS`	1	Flags to indicate the current I/O status of the slave MCU I/O • Bit 0: microprinter control (`1`: being executed). • Bit 1: external cassette read/write (`1`: being executed). • Bit 2: internal microcassette read/write (`1`: being executed). • Bit 3: RS-232C receive (`1`: being executed). • Bit 5: ROM cartridge power supply (`1`: ON). • Bit 6: bar-code reader cartridge power supply (`1`: ON). • Bit 7: `BREAK` (`1`: slave MCU I/O control forcibly terminated by master MCU).

Continues in next page...

...continued from previous page.

Address (from)	Address (to)	Variable name	Number of bytes	Description
`7D`	`7D`	`MIOSTS`	1	Indicates the I/O status of the master MCU. • Bit 0: read/write to LCD (`1`: being executed). • Bit 1: command transmit and response with slave MCU (`1`: being executed). • Bit 2: data communication using the external serial port (floppy disk unit) (`1`: being executed). • Bit 3: clock interrupt (alarm, square wave, update). (`1`: interrupt). • Bit 4: voltage low (`1`: voltage low interrupt). • Bit 5: power OFF (`1`: power switch interrupt). • Bit 6: PAUSE key ON (`1`: PAUSE key pressed). • Bit 7: BREAK key ON (`1`: BREAK key pressed).

Continues in next page...

...continued from previous page.

Address (from)	Address (to)	Variable name	Number of bytes	Description
7E	7E	SDIPS1	1	Software switch. • Bits 0 to 1: specify the type of waveform from the external cassette. Bit 0 / 1 0 / x Decided automatically. 1 / 0 Normal waveform. 1 / 1 Reverse waveform. • Bits 2 to 3: specify the type of waveform from the internal microcassette. Bit 2 / 3 0 / x Decided automatically. 1 / 0 Normal waveform. 1 / 1 Reverse waveform. • Bits 4 to 5: memory bank selection. • Bit 6: indicates the memory bank in which the BASIC interpreter is located (value is set when the menu is initialized). – 0: bank 0. – 1: bank 1. • Bit 7: Specifies access of addresses 0000 to 004D. – 0: access disabled. – 1: access enabled.

Continues in next page...

...continued from previous page.

Address (from)	Address (to)	Variable name	Number of bytes	Description

`7F`	`7F`	`SDIPS2`	1	Software switch.
				• Bits 0 to 3: correspond to DIP switches 1 to 4.
				  – `0`: OFF.
				  – `1`: ON.
				• Bit 4: flag indicating whether DIP switched 1 to 4 will be controlled by software (bits 0 to 3 above) or by the actual setting.
				  – `0`: actual DIP switch setting.
				  – `1`: bits 0 to 3.
				• Bit 5: flag indicating whether bit 7 will control the printer ON/OFF switch.
				  – `0`: actual printer ON/OFF switch setting.
				  – `1`: bit 7.
				• Bit 6: undefined.
				• Bit 7: controls the printer ON/OFF switch.
				  – `0`: OFF.
				  – `1`: ON.
				Note: these switches are included in the key matrix. The values of these switches are therefore set in the key matrix (`NEWKTB`) after key scanning.

Continues in next page...

...continued from previous page.				
Address (from)	(to)	Variable name	Number of bytes	Description

14.5 RAM system variables

Address (from)	(to)	Variable name	Number of bytes	Description
0100	0102	INTCLK	3	Address of real-time clock interrupt routine (for alarm, etc.) Address 0100 contains 7E (JMP command) and 0101, 0102 the upper and lower bytes of the jump address. Address values are initialized on reset.
0103	0105	INTEXT	3	Address of IRQ1 external port interrupt routine. Contents are identical to INTCLK.
0106	0108		3	Address of TRAP interrupt routine. Contents are identical to INTCLK.
0109	010B		3	Address of SCI interrupt routine. Contents are identical to INTCLK.
010C	010E	INTOF	3	Address of TOF interrupt routine. Contents are identical to INTCLK.
010F	0111		3	Address of OCF interrupt routine. Contents are identical to INTCLK.
0112	0114		3	Address of ICF interrupt routine. Contents are identical to INTCLK.
0115	0117		3	Address of IRQ1 interrupt routine. Contents are identical to INTCLK.
0118	011A	INTSW1	3	Address of the SW1 routine. Three bytes are reserved.
011B	011D		3	Address of the NMI routine. Three bytes are reserved.
011E	011F	FNTGPN	2	Address of the character fonts for codes E0-FF (upper- and lower-byte sequence).

Continues in next page...

...continued from previous page.

Address (from)	(to)	Variable name	Number of bytes	Description
0120	0121	BRKADR	2	Address of the subroutine to be executed when the BREAK key is pressed. This specification is valid only when RUNMOD is in machine language mode.
0122	0123	MENADR	2	Address of the subroutine to be executed when the MENU key is pressed. Contents are identical to BRKADR.
0124	0125	PAUADR	2	Address of the subroutine to be executed when the PAUSE key is pressed. Contents are identical to BRKADR.
0126	0127	CT3ADR	2	Address of the subroutine to be executed when the Ctrl+PF3 key is pressed. Control jumps unconditionally to this address. Address value is initialized at reset.
0128	0129		2	Address of the subroutine to be executed when the Ctrl+PF4 key is pressed. Contents are identical to CT3ADR.
012A	012B		2	Address of the subroutine to be executed when the Ctrl+PF5 key is pressed. Contents are identical to CT3ADR.
012C	012D	RMBADR	2	Shows the end of the RAM area. This variable is set when the RAM is checked at initialization (Ctrl+@ input from MENU)- Last address of the RAM + 1 is stored in upper- and lower-byte sequence.
012E	012F	PRMCNT	2	Address where the amount of data remaining in the PROM cartridge file data is stored.

Continues in next page...

...continued from previous page.				
Address (from)	(to)	Variable name	Number of bytes	Description
0130	0131	WAKADR	2	Address of the subroutine executed by the clock alarm interrupt at reset (power ON). Address is in upper- and lower-byte sequence. This address is initialized at reset.
0132	0133	POFADR	2	Address of the last subroutine called prior to turning OFF the power supply. Address is in upper- and lower-byte sequence. This address is initialized at reset.
0134	0135	BSWTAD	2	Starting address of the BASIC application area. Value of RMBADR is set at MENU initialization (Ctrl+@). Set to same value as RMBADR.
0136	0137	BSWBAD	2	Starting address of the BASIC program area.
0138	0139		2	Address of the BASIC work area save and condense routine.
013A	013A	BITMP0	1	Bank 0 bit map.
013B	013B	BITMP1	1	Bank 1 bit map.
013C	013F	LNKTBL	4	Address of the RAM application program link table.

14.6 RAM area used by I/O routines

Memory range	Description
004E to 007F	Flag and work area.
0100 to 0110	Interrupt entry pointer.
011E to 0139	Vector.
013A to 013F	Menu and link tables.
0140 to 018F	Keyboard work area.
Continues in next page...	

...continued from previous page.

Memory range	Description
`0190` to `01AE`	Microprinter work area.
`01AF` to `01C3`	RS-232C work area.
`01C4` to `01D5`	Serial communication work area.
`01D6` to `01EB`	External cassette work area.
`01EC` to `0207`	Built-in microcassette work area.
`0208` to `020E`	ROM cartridge work area.
`020F` to `021A`	Binary memory dump, memory load work area.
`021B` to `021F`	Reserved.
`0220` to `029F`	Screen (including LCD routine) routine work area.
`02A0` to `02CF`	Monitor work area.
`02D0` to `0323`	External cassette header work area.
`0324` to `0377`	Built-in microcassette work area.
`0380` to `047C`	Reserved for system buffer (260 bytes).

Chapter 15

Virtual screen

Note: this chapter contains descriptions related to the display controller for external display. This hardware is not available in every country. In countries where the display controller is not available,this chapter should be ignoring all references to the display controller and the external display.

15.1 General

The virtual screen is intended to allow the HX-20 to use a larger screen than the physical screen size of its LCD (20 columns by 4 lines). This function is good for both the LCD and the displayu controller (for external display).

The virtual screen has a maximum size of 255 columns by 255 lines. The display area where characters actually appear is called a “window” (the size of this window becomes 32 columns by 16 lines with the display controller). It functions as a viewing window through which any part of the large internal screen can be seen. The virtual screen on the LCD is controlled by the master MCU, whereas that of the external display is controlled by the display controller via a high-speed serial communication interface.

15.2 Names and technical terms

1. Virtual screen and physical screen

   Only character (or text) information is handled by the virtual screen. Its maximum size is 255 columns by 255 lines. For the LCD, a screen image is produced on the MCU memory. As opposed to the virtual screen, the screen used for actual display is called a “physical screen”.

The size of the physical screen is 20 columns by 4 lines for the LCD display and 32 columns by 16 lines for the display controller. Graphic display (straight line, etc.) is applicable to the physical screen only.

2. Window

   The window is a portion of the virtual screen that is actually displayed for viewing. The contents of the window are the same as those of the physical screen.

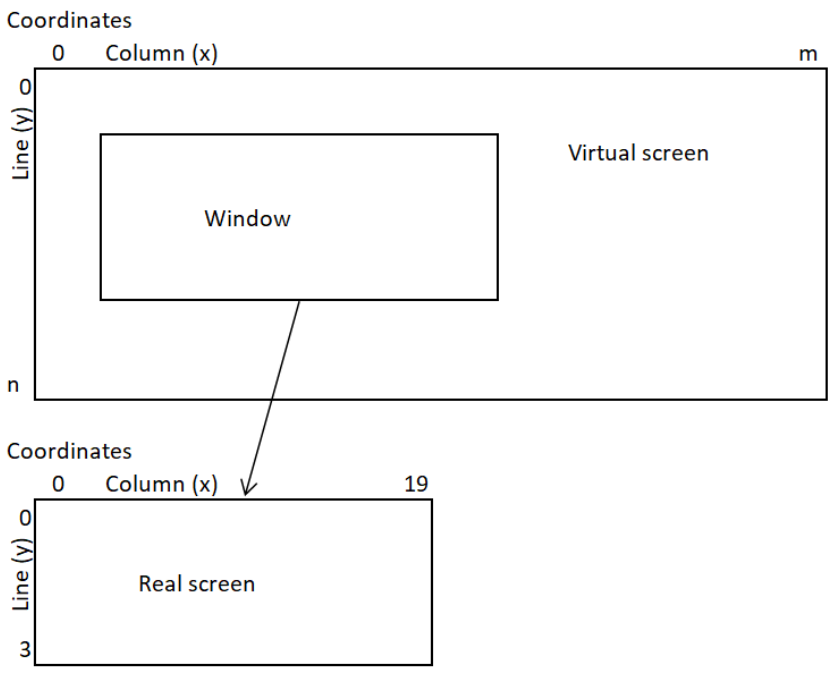


Figure 15.1: Virtual screen, physical screen and window

3. Coordinates on the screen

   The screen in a horizontal direction is called “columns”, while that in a vertical direction is called “lines”. Coordinates are represented by x and y, which correspond to columns and lines, respectively. Column 0 indicates the left end of the screen, while line 0 indicates the top of the screen. When the screen size is (m, n), the upper left end of the screen is identified by coordinates $(0, 0)$ and the lower right end by coordinates $(m - 1, n - 1)$.

4. Scroll

The scroll refers to the movement of the contents of the window up by one line (namely, the contents of the 4th line appear in the 3rd line, the contents of the 3rd line in the 2nd line and the contents of the 2nd line in the 1st line. New data appears in the 4th line. In the HX-20, this function is also applicable to the movement of the screen in the upward, left and right directions.

5. Scroll step

   A character code to specify the number of scroll steps. When this code is accepted, the screen scrolls by the number of columns or lines specified by this code.

6. Scroll of virtual screen

   The scroll of virtual screen refers to the movement of the contents of the virtual screen up or down by one line.

7. Line status

   In some cases, two lines of data to be displayed are desired to be handled as a single line. To support this, a flag is provided to indicate a continuation line for each line. This flag is called a "line status flag" (see Figure 15.2). The line status has a value "`FF`" if the line is a continuation of the preceding line and a value "`00`" if the line is a new line.

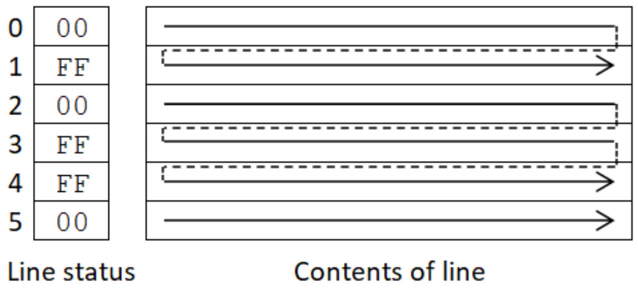


Figure 15.2: Line status

In Figure 15.2, 0th and 1st lines, 2nd through 4th lines and 5th line are logical single lines, respectively. The conditions for composing a logical single line are detailed in Section 15.4.

8. Cursor and cursor margin

   The cursor indicates the position of the character to be displayed. At the same time, it also serves as a reference point for screen control.

The cursor is designed to always stay within the window. If the cursor moves out of the window, the window also moves with the cursor. When the cursor is at either end of the window, the next character cannot be identified. Therefore, a certain width from either end of the window must be predetermined so that the window moves when the cursor reaches this position. This width is calles a “cursor margin”.

In the following example, the screen size of 40 columns by 8 lines has been defined for LCD display. Assume that the cursor margin is set to a value of 3 and the position of the right margin is “RM”, while that of the left margin is “LM”. When the cursor is between the positions “LM” and “RM” (i.e., the shaded section in Figure 15.3), window movement will not take place. When the cursor moves and reaches position “RM” (3rd position from the right), the cursor will not advance; instead the window will move to the right even if an attempt is made to move the cursor. This movement of the window stops when it reaches the right end of the virtual screen. From this point, the cursor moves again.

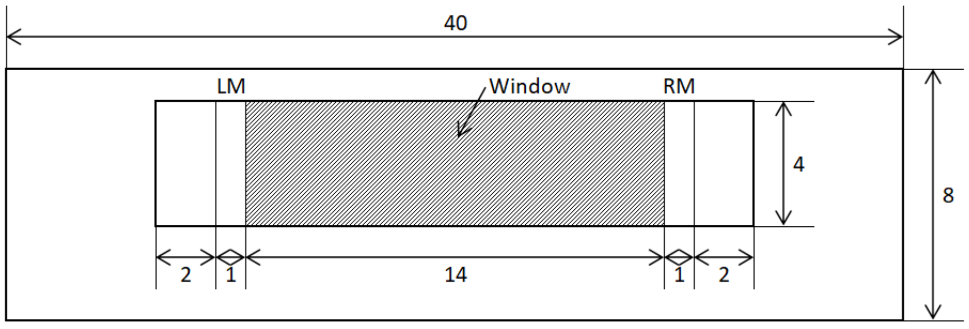


Figure 15.3: Cursor margin and window movement

The cursor margin may be specified by a value in the range of 1 to 10. If the value is 1, it indicates that no cursor margin is specified.

9. List flag

   If the window moves so that it contains the cursor, the displayed data is difficult to read. In some cases, the window may be desired to be fixed at the left end of the virtual screen (e.g., `LIST` command in BASIC). The list flag controls the movement of the window. When the list flag is set, the window moves along the left end of the virtual screen (see the shaded section in Figure 15.4).

   When the list flag is ON, the window cannot move horizontally. However, its vertical movement is not restricted.

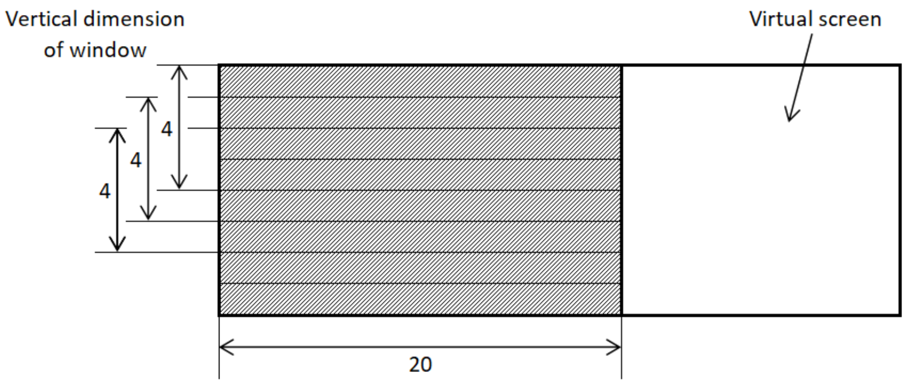


Figure 15.4: Moving range of window when list flag is ON

10. Access pointer

    When a character is to be input or output to or from the display controller, the location (i.e., coordinates on the virtual screen) where the character may be accessed for input/output must be specified. The access pointer functions to indicate this location.

15.3 Graphic display

Only character codes may be handled by the virtual screen. It cannot handle graphic data. Graphic data processing is supported by both the LCD and display controller but in a manner different from each other.

1. LCD

   Graphic data is processed only on the physical screen. Display functions such as dot ON/OFF, straight line drawing, etc. are controlled directly against the controller. Therefore, the contents of the virtual screen will not be lost even if the graphic display is activated.

2. Display controller

   On the display controller, both text and graphic data cannot be displayed concurrently. Therefore, either the mode to effect the display or that to effect the graphic display must be selected by changing the display mode. Moreover, because of the limited memory size of the display controller, the contents of the virtual screen will be lost when the graphic display is activated. The display controller is capable of color selection, which is different depending on the display mode. In text display mode, the background colors are green or orange with the color of all characters fixed. In graphic display mode, there are two color sets 0 and 1. All the colors in the same color set can be used as background colors. Other colors are available for dots.

Color set 0	Color set 1
Green	White
Yellow	Cyan
Blue	Magenta
Red	Orange

15.4 Virtual screen control

The movement of the cursor, deletion of one character, and other controls related to the display contents on the virtual screen are performed by using character codes. Special controls such as screen size specification, list control, etc., are provided as the functions of the virtual screen.

The character codes used are `00` through `FF`. Codes `20` through `FF` are those to be displayed on the screen as graphic characters. Codes `20` through `1F` are non-graphic characters which are not displayed on the screen. They are used as control characters for cursor movement, etc. The description of each character code follows.

1. Graphic characters

   (a) When not at the right-hand end on the bottom line

   The next line is assumed to be a continuation line (line status is `FF`, see Figure 15.5).

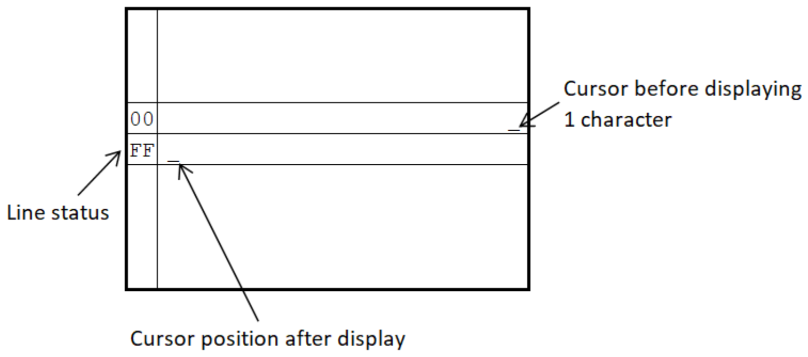

Cursor before displaying 1 character
00
FF
Line status
Cursor position after display

Figure 15.5: Continuation line

(b) When at the right-hand end on the bottom line

The display contents are scrolled up by one line. The bottom line becomes a continuation line filled with blank codes (`20`).

2. Control codes

19 character codes can be used as control codes. The functions of the respective control codes are as follows:

(a) `01` (window left)

Positions the window to the left of the virtual screen. The cursor moves to the 10th column of the window.

(b) `04` (scroll right)

Moves the window to the right by the number of columns specified by the horizontal scroll steps. However, the window will not move beyond the right end of the virtual screen.

(c) `05` (clear to end of line)

Changes all the characters from the cursor position to the end of the logical single line to blank codes (`20`).

(d) `06` (window right)

Positions the window to the right end of the virtual screen. The cursor moves to the 10th column of the window.

(e) `08` (delete one character)

Moves the cursor position back by one character and deletes the character at the cursor position. All the data following the deleted character on the line are shifted and a blank code (`20`) is entered at the end of the line. When the cursor is at the beginning of the line and therefore cannot be moved back, the character at the current cursor position is deleted.

(f) `09` (TAB)

Moves the cursor to the next tab position. Tab positions are set at every 8 columns such as 0, 8, 16,...

(g) `0A` (line feed)

Moves the cursor down by one line. When the cursor is at the bottom line of the virtual screen, the virtual screen scrolls one line and the bottom line will be filled with blank codes.

(h) `0B` (home)

Positions the cursor to the upper left corner of the virtual screen. The window moves along with the cursor (this position is referred to as "home position").

(i) `0C` (clear)

Changes all the contents of the virtual screen to blank codes (`20`). The logical single line is set to the virtual screen width and the cursor returns to the home position.

(j) `0D` (carriage return)

Terminates the logical single line (the line status of the next line becomes `00`). The cursor moves to the left end of the line.

(k) `10` (scroll up)

Moves the window up by the number of lines specified by the vertical scroll steps. The window will not move beyond the top end of the virtual screen. The cursor moves to the 10th column of the virtual screen.

(l) `11` (scroll down)

Moves the window down by the number of lines specified by the vertical scroll steps. The window will not move below the bottom end of the virtual screen. The cursor moves to the 10th column of the virtual screen.

(m) `12` (insert)

Inserts a blank code (`20`) into the cursor position. All the characters followinf the cursor position are shifted to the right by 1 column. If the last character in the logical single line is a blank code, that character is deleted. If the last character is not a blank code, another line filled with blank codes will be inserted (i.e., scrolling takes place above the cursor position) and the last character is positioned at the beginning of the inserted line.

(n) `13` (scroll left)

Moves the window to the left by the number of columns specified by the horizontal scroll steps. However, the window will not move beyond the left end of the virtual screen.

(o) `1A` (clear to end of screen)

Changes the contents of the virtual screen from the current cursor position to the end of the virtual screen with blank codes. The logical single line is set to the virtual screen width (line status is changed to "`00`").

(p) `1C` (cursor right)

Moves the cursor to the right by one column. The cursor at the right end of a line will move to the beginning of the next line. If the cursor is on the bottom line, it will move to the left end of the same line.

(q) `1D` (cursor left)

Moves the cursor to the left by one column. The cursor at the left end of a line will move to the right end of one immediately above the line. If the cursor is on the top line, it will move to the right end of the same line.

(r) `1E` (cursor up)

Moves the cursor up by one line. The cursor will not move if it is in the top line.

(s) `1F` (cursor down)

Moves the cursor down by one line. The cursor will not move if it is in the bottom line.

3. Subroutine call for virtual screen

The virtual screen is supported by subroutine "`SCRFNC`". Parameters for this subroutine are given by the parameter packet used on the memory. The packet begins with a 1-byte function code which is followed by a series of several data. The return information is also included in the packet.

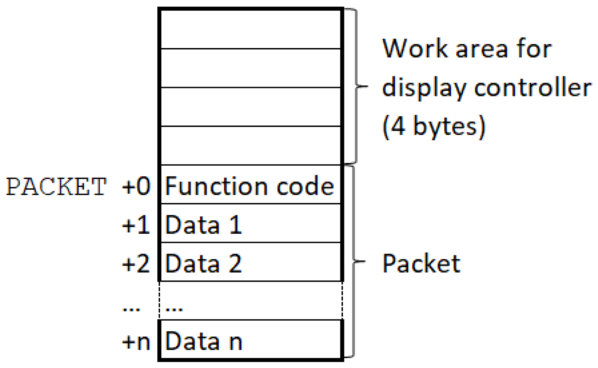


Figure 15.6: Parameter packet

A 4-byte area is required before the packet for the display controller functions (see Figure 15.6).

To call subroutine “SCRFNC” (entry point FF5E), the top address of the packet must be given to the index register.

Example: to call the function to set the virtual screen. In this example, the screen size of 40 × 8 is defined for the LCD and the buffer address is specified at 5000.

```
SCRFNC EQU   $FF5E
       LDX   #PACKET
       JSR   SCRFNC
;      ...
PACKET FCB   $87      ; Function code (define screen size)
       FCB   39       ; Screen width
       FCB   7        ; Screen depth
       FDB   $5000    ; Buffer address
;      ...
       ORG   $5000
       RMB   40*9+1   ; Buffer size
```

15.4.1 Functions for initialization of virtual screen

The following functions must be executed to initialize the virtual screen.

1. Function 84 (screen device select).
2. Function 87 (specification of screen size and buffer address).
3. Function C3 (specification of scroll margin).
4. Function C4 (specification of scroll steps).
5. Function CB (specification of scrolling speed).

Refer to Section 15.5 for detailed description of each function code.

15.5 Virtual screen function table

Packets for the virtual screen are listed below. The virtual screen functions are divided into those shared by both the LCD and the display controller and those peculiar to either one. The device to which the particular function is applicable is shown in the “Function (application)” column as (LCD) for the display and as (Disp) for the display controller. Data in each packet are numbered as 00, 01, 02,... for each byte and their descriptions are given in order of the data number. These packets are used at the time of both the entry and return of each subroutine. In the following table, "`XX`" indicates that arbitrary 2-digit values may be used, and unless otherwise specified, all numeric values are hexadecimal.

Function (Application)	Packet data number	Description (at entry)	Description (at return)
84		Screen device select.	
(Disp)	00	84 (function code)	
(LCD)	01	Device code • 30: display • 22: LCD	Return code • 00: normal. • FE: device is not connected. • FF: device specification is invalid.
85		Initialization of the display controller. The display controller is initialized at the default value.	
(Disp)	00	85 (function code)	
	01	XX	Return code • 00: normal. • FF: I/O error.
87		Specification of the virtual screen. By this function, the screen size and the top address of the buffer are specified. When the screen size is m columns by n lines, the size of the buffer must be $m \times n + 1$ bytes.	
(Disp)	00	87 (function code)	
(LCD)	01	Screen width (specify $m - 1$ for m columns).	Return code • 00: normal. • FF: screen oversize.
	02	Screen length (specify $n-1$ for n lines).	XX

Continues in next page...

...continued from previous page.

Function (Application)	Packet data number	Description (at entry)	(at return)
	03	High-order byte of buffer's top address.	XX
	04	Low-order byte of buffer's top address. **Note:** buffer addressing is not required if the display controller is specified.	XX
88		Input of the virtual screen size. By this function, the currently defined size of the virtual screen is obtained.	
(Disp)	00	88 (function code)	
(LCD)	01	XX	Screen width ($m-1$ for m columns).
(LCD)	02	XX	Screen length ($n-1$ for n lines).
89		Input of the window size.	
(Disp)	00	89 (function code)	
(LCD)	01	XX	Width: • 19_{10} for LCD. • 31_{10} for display controller.
	02	XX	Length: • 3_{10} for LCD. • 15_{10} for display controller.
8A		Input of the window position. By this function, the coordinate values at the upper left corner of the window on the virtual screen are given.	
(Disp)	00	8A (function code)	
(LCD)	01	XX	Coordinate x
	02	XX	Coordinate y

Continues in next page...

...continued from previous page.

Function (Application)	Packet data number	Description (at entry)	Description (at return)
8C		Input of the cursor position. By this function, the position of cursor on the virtual screen is obtained.	
(Disp)	00	8C (function code)	
(LCD)	01	XX	Coordinate x
	02	XX	Coordinate y
8D		Input of the cursor margin value.	
(Disp)	00	8D (function code)	
(LCD)	01	XX	Margin value.
8E		Input of the scroll steps.	
(Disp)	00	8E (function code)	
(LCD)	01	XX	Number of horizontal scroll steps.
	02	XX	Number of vertical scroll steps.
8F		By this function, the dot status at the specified position on the physical screen is obtained.	
(Disp)	00	8F (function code)	
(LCD)	01	High-order byte of coordinate x.	1. LCD • FF: ON. • 00: OFF. 2. Display controller: color code.
	02	Low-order byte of coordinate x.	XX
	03	High-order byte of coordinate y.	XX
	04	Low-order byte of coordinate y.	XX
91		Input of the range of the logical single line. By this function, the range of the logical single line containing the cursor on the virtual screen is obtained.	

Continues in next page...

...continued from previous page.

Function (Application)	Packet data number	Description (at entry)	(at return)
(Disp)	00	91 (function code)	
(LCD)	01	XX	First column in the logical single line (coordinate x with a value 0).
	02	XX	First line in the logical single line (coordinate y).
	03	XX	Physical screen width (LCD: 19_{10}; Disp: 31_{10}).
	04	XX	Last line in the logical single line (coordinate y).
92		Display of one character on the virtual screen.	
(Disp)	00	92 (function code)	
(LCD)	01	Character code	Coordinate x of the new cursor position.
	02	XX	Coordinate y of the new cursor position.
93		Specification of a display mode for the display controller.	
(Disp)	00	93 (function code)	
	01	Text mode • 00: graphic mode. • 01: text mode.	Return code • 00: normal. • FF: an error has occurred.

Continues in next page...

...continued from previous page.

Function (Application)	Packet data number	Description (at entry)	Description (at return)
	02	Graphic mode • `00`: text mode. • `01`: color graphic mode. • `02`: monochromatic graphic (high-resolution) mode. **Note:** text mode and graphic mode must be specified exclusively. In other words, either data `01` or `02` must be `00`. Graphical mode is supported on the physical screen. The resolution of the display is 128 × 64 in color graphic mode and 128 × 96 in monochromatic graphic mode (i.e., high-resolution mode).	`XX`
	03	Background color `00`: green `04`: white `01`: yellow `05`: cyan `02`: blue `06`: magenta `03`: red `07`: orange **Note:** background color selection is effective in graphic mode only. A color set is defined by the `COLOR` command in text mode.	`XX`
`95`		Input of one character on the display. By this command, the character at the coordinates specified by the access pointer is input.	
(Disp)	`00`	`95` (function code)	

Continues in next page...

...continued from previous page.

Function (Application)	Packet data number	Description (at entry)	(at return)
	01	XX	Character code.
	02	XX	Color code (background color code).
97		Consecutive input of characters from the virtual screen. By this function, characters are input in the number specified from the coordinate positions where data read starts.	
(Disp)	00	97 (function code)	
(LCD)	01	Coordinate x at the read start point.	Input character 1.
	02	Coordinate y at the read start point.	Input character 2.
	03	Number of read characters.	Input character 3.
	04	XX	
98		Display of one character on the virtual screen (note that the packet generation in this case is different from that of function 92).	
(Disp)	00	98 (function code)	
(LCD)	01	XX	Coordinate x of the new cursor position.
	02	XX	Coordinate y of the new cursor position.
	03	XX	First line number in the logical single line containing the new cursor (coordinate y).
	04	XX	Last line number in the logical single line containing the new cursor (coordinate y).
C0		Setting of the window position. By this function, the upper left edge of the window is positioned at the specified address on the virtual screen.	
(Disp)	00	C0 (function code)	

Continues in next page...

…continued from previous page.

Function (Application)	Packet data number	Description (at entry)	Description (at return)
(LCD)	`01`	Coordinate x on the virtual screen.	`XX`
	`02`	Coordinate y on the virtual screen. **Note:** if the window position is outside the bounds of the virtual screen, the maximum values are set for both coordinates x and y.	`XX`
`C2`		Specification of the cursor position. By this function, the cursor is placed at the specified position on the virtual screen, resulting in the movement of the window.	
(Disp)	`00`	`C2` (function code)	
(LCD)	`01`	Coordinate x of the cursor position.	`XX`

Continues in next page…

...continued from previous page.

Function (Application)	Packet data number	Description (at entry)	Description (at return)
	`02`	Coordinate y of the cursor position. **Note:** the window movement is controlled as follows: 1. The window does not move when the specified cursor position is within the window area. 2. When the specified position is not within the window area, the window moves so that the new cursor is located at the home position of the window. The cursor position cannot be located at the home position of the window, if the bottom edge of the window is in alignment with the bottom edge of the virtual screen. In such a case, the cursor position is set within the window area according to the same rule as that of function code `C0`.	`XX`
`C3` (Disp) (LCD)		Setting of the value of the cursor margin.	
	`00`	`C3` (function code)	
	`01`	Cursor margin value (this value must be in the range from 1 to half the window value).	`XX`
`C4` (Disp)		Setting of the number of scroll steps.	
	`00`	`C4` (function code)	

Continues in next page...

...continued from previous page.

Function (Application)	Packet data number	Description (at entry)	(at return)
(LCD)	`01`	Number of horizontal scroll steps (0 to 255_{10}).	`XX`
	`02`	Number of vertical scroll steps (0 to 255_{10}).	`XX`
`C5`		Turning the list flag ON.	
(Disp) (LCD)	`00`	`C5` (function code)	
`C6`		Resetting of the list flag.	
(Disp) (LCD)	`00`	`C6` (function code)	
`C7`		Setting of a dot at the specified position. This function id effective in graphic mode.	
(Disp) (LCD)	`00`	`C7` (function code)	
	`01`	High-order byte of coordinate x.	`XX`
	`02`	Low-order byte of coordinate x.	`XX`
	`03`	High-order byte of coordinate y.	`XX`
	`04`	Low-order byte of coordinate y.	`XX`

Continues in next page...

...continued from previous page.

Function (Application)	Packet data number	Description (at entry)	Description (at return)

	05	Color code.
		• With LCD
		  – `00`: OFF.
		  – `FF`: ON.
		• With display controller
		  – If color set 0 is specified
		    ∗ `00`: green.
		    ∗ `01`: yellow.
		    ∗ `02`: blue.
		    ∗ `03`: red.
		  – If color set 1 is specified
		    ∗ `00`: white.
		    ∗ `01`: cyan.
		    ∗ `02`: magenta.
		    ∗ `03`: orange.
			`XX`

`C8`		Drawing a straight line between any two points on the graphic screen.	
(Disp) (LCD)	`00`	`C8` (function code)	
	`01`	High-order byte of coordinate x at the start point.	`XX`
	`02`	Low-order byte of coordinate x at start point.	`XX`

Continues in next page...

...continued from previous page.			
Function (Application)	Packet data number	Description	
		(at entry)	(at return)
	03	High-order byte of coordinate y at start point.	XX
	04	Low-order byte of coordinate y at start point.	XX
	05	High-order byte of coordinate x at the end point.	XX
	06	Low-order byte of coordinate x at end point.	XX
	07	High-order byte of coordinate y at end point.	XX
	08	Low-order byte of coordinate y at end point.	XX
	09	Color code. Same as function code C7.	XX
C9		Termination of the logical single line. By this function, the line status of the specified line is reset to 00.	
(Disp)	00	C9 (function code)	
(LCD)	01	Line number (coordinate y).	XX
CA		Clearing of the screen in graphic mode. This function is effective for the graphic screen when the display controller is used, and for the physical screen when the LCD display is used.	
(Disp)	00	CA (function code)	
	01	Background color (effective only with the display controller).	XX
CB		Setting of the scrolling speed. This function specifies the scrolling speed of the physical screen.	
(LCD)	00	CB (function code)	
Continues in next page...			

...continued from previous page.

Function (Application)	Packet data number	Description (at entry)	Description (at return)
	01	Speed. A value in the range of `00` to `09` is used to specifiy the scrolling speed. 9 is the highest scrolling speed.	`XX`
`CD`		Output of one character to the position specified by the access pointer.	
(Disp)	00	`CD` (function code)	
	01	Character code.	`XX`
`CE`		Specification of the access pointer. By this function, the character position that can read/write on the virtual screen when the display controller is used is specified.	
(Disp)	00	`CE` (function code)	
	01	Coordinate x of the access pointer.	`XX`
	02	Coordinate y of the access pointer.	`XX`
`CF`		Specification of a color set. Two color sets each consisting of 4 different colors are selectable when the display controller is used.	
(Disp)	00	`CF` (function code)	
	01	Color set. • `00`: color set 0. • `01`: color set 1. If color set 0 is specified, green, yellow, blue and red can be used. If color set 1 is specified, white, cyan, magenta and orange can be used.	`XX`

15.6 EPSP message format table for screen

In the following table, SS and MM refer to the slave and master device numbers, respectively. Numeric values are all hexadecimal. "XX" indicates that arbitrary 2-digit values may be used.

Function code	FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
84							Screen device select.
	00	SS	MM	84	00	00	Device number (30).
	01	MM	SS	84	00	00	Return code. • 00: normal. • FE: device is not ready. • FF: device number is invalid.
85							Initialization of screen.
	00	SS	MM	85	00	00	XX
	01	MM	SS	85	00	00	Return code. • 00: normal. • FF: an error has occurred.
87							Specification of the screen size.
	00	SS	MM	87	03	00	Virtual screen width (maximum value of coordinate x).
						01	Virtual screen length (maximum value of coordinate y).
						02	XX
						03	XX

Continues in next page page...

...continued from previous page.

Function code	FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
	01	MM	SS	85	00	00	Return code. • 00: normal. • FF: size specification is invalid.
88							Input of the virtual screen size.
	00	SS	MM	88	00	00	XX
	01	MM	SS	88	01	00	Virtual screen width (maximum value of coordinate x).
						01	Virtual screen length (maximum value of coordinate y).
89							Input of the physical screen.
	00	SS	MM	89	00	00	XX
	01	MM	SS	89	01	00	Screen width (maximum value of coordinate x).
						01	Screen length (maximum value of coordinate y).
C0							Positioning of the physical screen on the virtual screen. Position values are given with respect to the position $(0, 0)$ of the physical screen.
	00	SS	MM	C0	01	00	Coordinate x of the specified position.
						01	Coordinate y of the specified position.
	01	MM	SS	C0	00	00	XX

Continues in next page page...

...continued from previous page.

Function code	FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
8A							Input of the physical screen position on the virtual screen. Position values are given with respect to the position $(0, 0)$ of the physical screen.
	00	SS	MM	8A	00	00	XX
	01	MM	SS	8A	01	00	Coordinate x of the specified position.
						01	Coordinate y of the specified position.
C2							Specification of the cursor position on the virtual screen.
	00	SS	MM	C2	01	00	Coordinate x of the specified position.
						01	Coordinate y of the specified position.
	01	MM	SS	C2	00	00	XX
8C							Input of the cursor position on the virtual screen.
	00	SS	MM	8C	00	00	XX
	01	MM	SS	8C	01	00	Coordinate x of the specified position.
						01	Coordinate y of the specified position.
C3							Setting of the margin value of the cursor.
	00	SS	MM	C3	00	00	Margin value.
	01	MM	SS	C3	00	00	XX
8D							Input of the cursor margin value.
	00	SS	MM	8D	00	00	XX
	01	MM	SS	8D	00	00	Margin value.

Continues in next page page...

...continued from previous page.

Function code	FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
C4							Setting of the number of scroll steps.
	00	SS	MM	C4	01	00	Number of horizontal scroll steps.
						01	Number of vertical scroll steps.
	01	MM	SS	C4	00	00	XX
8E							Input of scroll steps.
	00	SS	MM	8E	00	00	XX
	01	MM	SS	8E	01	00	Number of horizontal scroll steps.
						01	Number of vertical scroll steps.
C5							Setting of the list flag.
	00	SS	MM	C5	01	00	XX
	01	MM	SS	C5	00	00	XX
C6							Resetting of the list flag.
	00	SS	MM	C6	01	00	XX
	01	MM	SS	C6	00	00	XX
C7							Setting of a dot at the specified position.
	00	SS	MM	C7	04	00	High-order byte of coordinate x.
						01	Low-order byte of coordinate x.
						02	High-order byte of coordinate y.
						03	Low-order byte of coordinate y.
						04	Color code.
	01	MM	SS	C7	00	00	XX
8F							Input of the dot status at the specified position.

Continues in next page page...

...continued from previous page.

Function code	FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
	00	SS	MM	8F	03	00	High-order byte of coordinate x.
						01	Low-order byte of coordinate x.
						02	High-order byte of coordinate y.
						03	Low-order byte of coordinate y.
	01	MM	SS	8F	00	00	Color code.
C8							Drawing of a straight line.
	00	SS	MM	C8	08	00	High-order byte of coordinate x at the start point.
						01	Low-order byte of coordinate x at the start point.
						02	High-order byte of coordinate y at the start point.
						03	Low-order byte of coordinate y at the start point.
						04	High-order byte of coordinate x at the end point.
						05	Low-order byte of coordinate x at the end point.
						06	High-order byte of coordinate y at the end point.
						07	Low-order byte of coordinate y at the end point.
						08	Color code.

Continues in next page page...

...continued from previous page.

Function code	FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
	01	MM	SS	C8	00	00	XX
91							Input of the range of the logical single line containing the cursor.
	00	SS	MM	91	00	00	XX
	01	MM	SS	91	03	00	00.
						01	Coordinate y of the first line in the logical single line.
						02	Column size of the physical screen.
						03	Coordinate y of the last line in the logical single line.
C9							Resetting of the line status of the specified line (i.e., partitioning of the logical single line).
	00	SS	MM	C9	00	00	Line number.
	01	MM	SS	C9	00	00	XX
92							Display of one character on the virtual screen.
	00	SS	MM	92	00	00	Character code.
	01	MM	SS	92	01	00	Coordinate x of the cursor.
						01	Coordinate x of the cursor.
CA							Specification of the background color in graphic mode.
	00	SS	MM	CA	00	00	Color code.
	01	MM	SS	CA	00	00	XX
CB							Setting of the scrolling speed.

Continues in next page page...

...continued from previous page.

Function code	FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
	00	SS	MM	CB	00	00	A value in the range of 0 to 9 is used to specify the scroll.
	01	MM	SS	CB	00	00	XX
93							Specification of display mode.
	00	SS	MM	93	02	00	Text mode. • 00: mode other than text mode. • 01: text mode.
						01	Graphic mode. • 00: mode other than graphic mode. • 01: graphic mode 1. • 02: graphic mode 2. **Note:** either data 00 or data 01 must be "00". Both data must not be "00".
						02	Background color. 00: green 01: yellow 02: blue 03: red 04: white 05: cyan 06: magenta 07: orange

Continues in next page page...

...continued from previous page.

Function code	FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
	01	MM	SS	93	00	00	Return code. • 00: normal. • FF: an error has occurred.
CD							Writing of one character into the access pointer.
	00	SS	MM	CD	01	00 01	Character code. Color code.
	01	MM	SS	CD	00	00	XX
CE							Specification of the access pointer against the virtual screen.
	00	SS	MM	CE	01	00 01	Coordinate x. Coordinate y.
	01	MM	SS	CE	00	00	XX
95							Input of one character from the access pointer.
	00	SS	MM	95	00	00	XX.
	01	MM	SS	95	01	00 01	Character code. Color code.
							Selection of a color set.
CF	00	SS	MM	CF	00	00	• 00: color set 0. • 01: color set 1.
	01	MM	SS	CF	00	00	XX
97							Input of characters at the positions specified consecutively on the virtual screen.

Continues in next page page...

...continued from previous page.							
Function code	FMT	DID	SID	FNC	SIZ	Text data no.	Description of function and text
	00	SS	MM	97	03	00	Coordinate x of the start point.
						01	Coordinate y of the start point.
						02	High-order byte of the number of input characters.
						03	Low-order byte of the number of input characters.
	01	MM	SS	97	*mm*	00 ... *mm*	00-*mm* denote the character codes of the input characters.
98							Display of one character on the virtual screen followed by the input of the first and last line numbers of the logical single line including the newly set cursor position.
	00	SS	MM	98	00	00	Character code.
	01	MM	SS	98	03	00	Coordinate x of the new cursor position.
						01	Coordinate y of the new cursor position.
						02	First line number in the single logical line.
						03	Last line number in the single logical line.

Table 15.2: Function codes for display controller.

Chapter 16

Menu

16.1 General

The title or entry point of a program can be registered or displayed by the MENU function of the HX-20. This chapter first describes the `ID` structure of the application programs stored in the ROMs of the HX-20, then explains how the `ID` information is displayed by the MENU with examples.

16.2 `ID` structure

The `ID` (identifying information also called a "header") for both the ROM ans user (RAM) application programs is structured as described below.

When the user writes an application program into a ROM or RAM and wishes to display the program on the MENU, he or she must write the header information to identify the program. Particularly, for an application program stored in a ROM, this header information must be at the top address (low-order address) of the ROM.

16.2.1 Header of ROM/RAM application program

1. `ID1` (1 byte)
    - Bit 0 - bit 6 '`:`' (code `3A`)
    - Bit 7
        - `0`: the header contains a link address to the next program on the ROM or RAM. The linked program is not displayed on the MENU. This bit can be used when the user writes programs using an EPROM. In other words, if the user wishes to erase a

program on the EPROM, bit 7 should be changed from logic "`1`" to "`0`" using an EPROM writer (bit 7 is "`1`" with the EPROM in the initialized state).

  - `1`: header contains a link address with the next program on the ROM or RAM, the starting address (entry point) of a program and its program name.

2. `ID2` (1 byte)

   - Bit 0 - bit 6: the header information contains one of the following codes:
     - "`A`": application program (application for general use).
     - "`B`": BASIC interpreter.
     - "`E`": end of link (no application program follows this header information).
       RAM application for general use.
   - Bit 7
     - `0`: indicates that the linkage with the next program is an absolute value (i.e., absolute address).
     - `1`: indicates that the linkega with the next program is a relative value (i.e., offset value from the header).
       If the ROMs are made available for use on any sockets, programs are relocatable and thus bit 7 must be set to logic "`1`".

3. Pointer to next header (2 bytes)

   This header information is also called a "link address".

   This two-byte data is used as a pointer to the location of the next header. If no header exists within the same ROM, the value of this data is "`FFFF`".

   If the MENU finds value "`FFFF`" on a ROM, it scans the next ROM for header information.

4. Starting address of program (entry point) (2 bytes)

   This header information indicates the starting address of a program. The starting address is an absolute value if the bit 7 of `ID2` is logic "`1`" and an offset value from the beginning of this header information if bit 7 is "`0`".

5. Filename (program name) (17 bytes max.)

   A filename is entered in a maximum of 16 bytes in ASCII code.

   The last byte of this header information is always "00".

16.2.2 Header of BASIC application programs

The header of a BASIC application program (i.e., an application program written in BASIC by the user) is different from that of a ROM/RAM application program.

BASIC application programs have no linkage with ROM/RAM application programs. However ROM/RAM application programs are displayed automatically by the MENU function.

1. Link offset (2 bytes)

   This is a pointer to indicate the starting address of the header of the next BASIC program. For example, program 1 points at program 2, while program 2 points to program 3. When the link offset value is FFFF, it indicates that no next header exists.

2. Filename (program name) (8 bytes)

   The filename of a program is specified by the TITLE command of BASIC. If the program has no filename, blanks must be entered as the filename in the header.

16.2.3 Bit map (2 bytes) and link tables (4 bytes, 013C to 013F

After the input of "Ctrl+@", the MENU generates a bit map which indicates the presence of the header of a ROM application program, and a link table for linkage with a RAM application program. Bit map addresses are 013A and 013B.

013A	Bit 7	ROM at addresses E000 to FFFF of bank 0.
	Bit 6	ROM at addresses C000 to DFFF of bank 0.
	Bit 5	ROM at addresses A000 to BFFF of bank 0.
	Bit 4	ROM at addresses 8000 to 9FFF of bank 0.
	Bit 3	ROM at addresses 6000 to 7FFF of bank 0.
	Bit 2	ROM at addresses 4000 to 5FFF of bank 0.
	Bit 1	ROM at addresses 2000 to 3FFF of bank 0.
	Bit 0	ROM at addresses 0000 to 1FFF of bank 0.

013B	Bit 7	ROM at addresses E000 to FFFF of bank 1.
	Bit 6	ROM at addresses C000 to DFFF of bank 1.
	Bit 5	ROM at addresses A000 to BFFF of bank 1.
	Bit 4	ROM at addresses 8000 to 9FFF of bank 1.
	Bit 3	ROM at addresses 6000 to 7FFF of bank 1.
	Bit 2	ROM at addresses 4000 to 5FFF of bank 1.
	Bit 1	ROM at addresses 2000 to 3FFF of bank 1.
	Bit 0	ROM at addresses 0000 to 1FFF of bank 1.

0: no header exists in the specified ROM socket.
1: header exists in the specified ROM socket.
Bank 0: main memory of HX-20.
Bank 1: memory in the expansion unit for HX-20.
The link table after the input of "Ctrl+@" contains 4-byte data:
"1:/'E'/FF/FF"

If the user wishes to display any program on the RAM in the MENU, he or she just needs to link this 4-byte data in the link table to his or her object program. For example, if the user writes an application program from address 1000, the header of the RAM application program and its link table should be written as follows:
1000/:(bit7=1)/'A'/FF/FF/10/20/U/S/E/R/00/
013C/:(bit7=0)/'A'/10/00/

16.2.4 How bit map and link table are generated

Neither a bit map nor a link table exists before the HX-20 system is initialized (by pressing Ctrl and @ keys, see Section 1.1.2 in BASIC Reference Manual). Before the system is cold started by "Ctrl+@", "CTRL/@ Initialize", "1 MONITOR" and dummy names (19 max.) will appear in the MENU on the LCD. After pressing Ctrl and @ keys, the MENU generates a bit map and a link table. When generating a bit map by the MENU, program linking starts from address D000 (MONITOR). Next, scanning of addresses starts from A000 (bank 1 also if an expansion unit is connected) and progresses to addresses 8000, 6000 and 4000 in the order named. The MENU sets the bit map depending on whether or not the header of an application program exists, and writes ":/'E'/FF/FF" into the link table.

Subsequently, the MENU displays the filename of a ROM application filename according to the bit map. Next, if there is any linked RAM (user) application program, then the name of the RAM application program is displayed, followed by BASIC application programs.

16.3 Examples

Address	Bank 0	Bank 1
0000		
1000	`BA 'A' FF FF 10 20 'USER-A' 00`	
2000		
4000		`BA 'A' 50 00 40 18 'APLC-5' 00`
5000		`BA 'A' FF FF 50 25 'APLC-4' 00`
6000	`BA 'A' FF FF 60 20 'APLC-2' 00`	
8000	`BA 'B' FF FF 80 10 'BASIC' 00`	`BA 'A' FF FF 80 33 'APLC-3' 00`
A000		
C000		
D000	`BA 'A' FF FF D0 33 'MONITOR' 00`	
E000		
FFFF		

Assume that there are 2 BASIC application programs (`APLC-1` and `APLC-2`) in addition to the above ROM/RAM application program.

The bit map in this case will be as follows:

	MSB LSB
13A	01011000
13B	00010100

and the link table will be as follows:

```
013C/:/'A'/10/00
```

The following information will appear in the MENU on the LCD display

```
CTRL/@ Initialize
1 MONITOR
2 BASIC
3 APLC-3
4 APLC-2
5 APLC-5
6 APLC-4
7 USER-A
8 APLC-1
9 APLC-2
```

16.4 MENU work areas

Address (from)	Address (to)	Variable name	Byte count	Description
2D0	48A	SCNBUF	442	Buffer for MENU display.
78	78	INTFLG	1	Initialize flag (0: request; 1: complete). • Bit 0: MENU • Bit 7: BASIC Condense (garbage collection) flag (1: condense request). • Bit 6: (BASIC, application) condense.
7B	7B	RUNMOD	1	Run mode. 01: MENU
7E	7E	SFTSWH	1	Software switch 1. • Bit 4: bank switch number currently selected (0: bank 0; 1: bank 1). • Bit 5: bank switch number selected before current number. • Bit 6: bank number in which BASIC programs are stored. Bits 5 and 6 are used to condense application.
80	81	TMPBF1	2	Temporary buffer.
82	83	CNTMNU	2	Indicates the top address of ROM (C000, A000, 8000,...).
84	84	CNTMNU	1	Number of items currently on the MENU display −1.
85	85	MNUNUB	1	MENU number.
86	86	BITMP	1	Bit map value of a bank (for temporary use).
87	87	BBTMP0	1	Buffer for BITMP0 (bit map of bank 0).
88	88	BBTMP1	1	Buffer for BITMP1 (bit map of bank 1).
89	89	STKLIN	1	Maximum number of lines on MENU display.

Continues in next page...

...continued from previous page.

Address (from)	(to)	Variable name	Byte count	Description
8A	8A	MXMNUB	1	Maximum number of MENUs (ASCII code).
8B	8B	BSAPNB	1	BASIC application number.
8C	8C	CNTFLG	1	Work area for temporary use.
8D	8D	DISFLG	1	Work area for temporary use.
8E	92	PCKT	5	LCD buffer work area for virtual screen packet.
13A	13A	BITMP0	1	Bit map for bank 0. Indicates whether the header of a ROM application program exists in one of the ROM chips in bank 0 (0: no header exists; 1: a header exists). Bit 0: Address 0000 of bank 0. Bit 1: Address 2000 of bank 0. Bit 2: Address 4000 of bank 0. Bit 3: Address 6000 of bank 0. Bit 4: Address 8000 of bank 0. Bit 5: Address A000 of bank 0. Bit 6: Address C000 of bank 0. Bit 7: Address E000 of bank 0.
13B	13B	BITMP1	1	Bit map for bank 1. Indicates whether the header of a ROM application program exists in one of the ROM chips in bank 1 (0: no header exists; 1: a header exists). Bit 0: Address 0000 of bank 1. Bit 1: Address 2000 of bank 1. Bit 2: Address 4000 of bank 1. Bit 3: Address 6000 of bank 1. Bit 4: Address 8000 of bank 1. Bit 5: Address A000 of bank 1. Bit 6: Address C000 of bank 1. Bit 7: Address E000 of bank 1.

Continues in next page...

...continued from previous page.

Address (from)	Address (to)	Variable name	Byte count	Description
`13C`	`140`	`LNKTBL`	4	Link table for RAM application programs. 1. When RAM application program does not exist. `: E FF FF` 2. When a header of a RAM application program exists `: A` *[address of the RAM application program]*

Chapter 17

Monitor

17.1 General

The Monitor is located in the ROM (ROM2) area from `C000` to `DFFF` and has two entry points `DFF7-DFF9` and `DFFA-DFFC`. The former id for entry from the menu display, etc., while the latter is for entry when a trap interrupt is generated. If one of the trap interrupt addresses (`0106` through `0108`) is specified, the default assumption is "`JMP $DFFA`". The display of data by the Monitor is always on the physical screen and the virtual screen is never used for the monitor display.

The HX-20 Monitor has 10 types of commands as listed below.

1. S (Set) command: displays and changes the contents of the memory.

2. D (Dump) command: displays the contents of the memory.

3. G (Go) command: executes a program.

4. X (eXamine) command: displays and changes the contents of each register.

5. R (Read) command: loads a program or data into the memory from an external storage.

6. W (Write) command: saves the contents of the memory to an external storage.

7. V (Verify) command: verifies the data output to an external storage.

8. K (Key) command: specifies the data for automatic key input when the power switch is turned ON.

9. A (Address) command: specifies the range of the memory space when loading from an external storage or saving data to an external storage.
10. B (Back) command: returns control to the procedure by which the Monitor was called.

Refer to Chapter 9 of the HX-20 Operation Manual for detailed description of each monitor command.

17.2 About Trap

If an attempt to execute a command not defined for the MCU is made, a trap interrupt is generated. By utilizing this characteristic, a breakpoint is set by the G command. For example, write "00" (undefined code) in the address specified as a breakpoint. Then try to execute the command at this address, and a trap interrupt will be generated, causing the HX-20 to return to the Monitor mode again.

Address		Variable	Byte	Description
(from)	(to)	name	count	
2A0	2A1	BP1	2	Stores the address specified as a breakpoint.
2A2	2A2	BPD1	1	Stores the contents of the breakpoint address.
2A3	2A3	LCDSTS	1	Stores the LCD status ('DISSTS': address 0280) when the HX-20 enters Monitor mode.
2A4	2BE		27	Work area for packets of binary dump/load routine.
2BF	2C0	PC	2	Stores the program counter value.
2C1	2C2	RTNADD	2	Stores return address on execution of B command.
2C3	2C4	LINLST	2	stores the buffer address corresponding to the end of the first line of the physical screen.
2C5	2C5	SRNMOD	1	Stores the R option of R command.
2C6	2CF		10	Unused.

Chapter 18

Interfacing with BASIC

18.1 Interfacing with sequential access devices

18.1.1 DCB (Device Control Blocks)

To perform I/O operations with sequential access devices such as cassette tapes, etc., a DCB is necessary to specify the conditions for interfacing. DCBs are required for each type of sequential access device (“`CAS0:`”, “`COM0:`”, etc.). The contents of the DCBs are shown below.

Item	Data No. (Size)	Description
1	`0-3` (4 bytes)	Device name (ASCII code). The four-character device name specified in the file descriptor is entered here.
2	`4` (1 byte)	I/O mode. Specified as one of the following values • `10`$_{16}$: sequential input. • `20`$_{16}$: sequential output. • `30`$_{16}$: sequential input/output.
3	`5-6` (2 bytes)	Entry point for the `OPEN` routine. The mode of the file (`10`$_{16}$: input, `20`$_{16}$: output) is stored int variable `FILMOD` (address `068A`). The `OPEN` routine references the mode data and opens the file for input or output.
		Continues in next page...

...continued from previous page.

Item	Data No. (Size)	Description
4	7-8 (2 bytes)	Entry point for the CLOSE routine. The CLOSE routine also references variable FILMOD and performs close for input or output.
5	9-10 (2 bytes)	Entry point for the input routine for one byte. The input routine inputs one byte is then stored in accumulator A. When the end of the file is detected, FF is entered in variable EOFFLG (address 00F8).
6	11-12 (2 bytes)	Entry point for the output routine for one byte. This routine outputs the contents of accumulator A.
7	13-14 (2 bytes)	Entry point for EOF routine. This routine sets data FF in accumulator B if the EOF is detected during input. Otherwise, 00 is entered in accumulator B.
8	15-16 (2 bytes)	Entry point for LOF routine. This routine enters the number of characters in the buffer or the remaining characters in the file in register D (accumulators A, B).
9	17-20 (4 bytes)	Reserved for data unique to each device.
10	21 (1 byte)	Specifies the column position of the next character to be output (leftmost column is taken to be column '0'). This value is returned when the POS function is called. Normally, this value is initialized to 0 and incremented by 1 each time one byte is output by the output routine. Reset to 0 by CR (code 0D) or LF (code 0A). When this value exceeds the range for the length of one line, and the next character is not CR or LF, the output routine for one byte automatically generates CR or LF and resets the column position to 0.

Continues in next page...

...continued from previous page.

Item	Data No. (Size)	Description
11	22 (1 byte)	Maximum value of characters per line. May be specified in the range `00` to `FF`. `00` indicates that the number of characters per line is infinite. As a result, BASIC does not automatically output `CR/LF`. `00` is set by executing `WIDTH` [*device name*]`, 255`.
12	23 (1 byte)	Specifies the size of the print zone when items in a `PRINT` statement are delimited by "," (comma). The default value is 14.
13	24 (1 byte)	Column position of last print zone. This value is according to the maximum number of characters in the line and the size of the print zone. For example, when the maximum number of characters in the line is 80 and the size of the print zone is 14, this value will be 56.
14	25 (1 byte)	If the number of characters per line can be changed with the `WIDTH` statement, `00` is entered as the value of this item. Otherwise, 80_{16} is entered.

18.1.2 DCB table

This is a 32-byte table which stores the addresses of the DCBs for each device. Addresses are specified in two bytes and up to 16 DCB addresses can be stored in this table. In the current version, seven addresses are stored in the DCB table and space for nine more addresses is reserved. Device numbers ($0 - 15_{10}$) are assigned to the DCBs in sequence. The variable name for the DCB table is `DCBTAB` (address `0657`).

18.1.3 Error processing

When an I/O error occurs during the execution of a routine or when the required device is busy, the corresponding error code is set in accumulator `B` and the followinf procedure is executed.

```
ERROR  EQU   $8433
       LDAB  #XX        ; Set the error code
       JMP   ERROR      ; Jump to the error handler
```

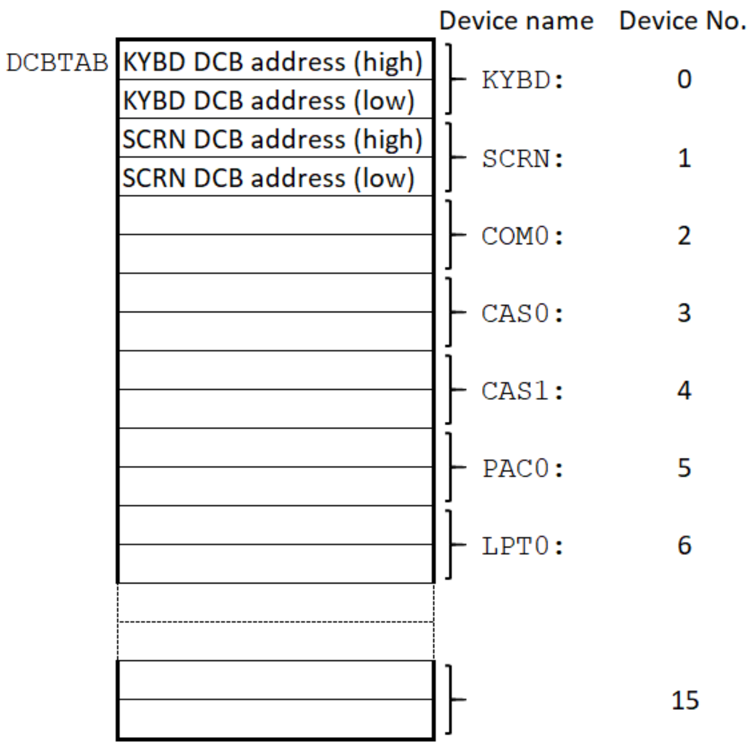


Figure 18.1: DCB table

The following error codes are commonly used.

Error code	Message	Description
53_{10}	`IO`	Error in communication with peripheral device.
59_{10}	`IU`	Specified device is in use (busy).
60_{10}	`DU`	Device is unavailable.

18.1.4 BREAK key processing

The following two procedures are available when `BREAK` signal is detected during execution of an I/O operation with a peripheral device.

1. Processing `BREAK` as an error

   In this case, processing is identical to that for an I/O error. Error code 53 (I/O error) is set in accumulator `B` and control is transferred to the error handling subroutine (label name `ERROR`).

```
LDAB  #53    ; Error code for I/O error
```

```
            JMP    ERROR
```

This procedure does not affect the other open devices or variables. When an `ON ERROR GOTO` statement has not been executed in the program mode, or when the I/O error occurs in the direct mode, the following error message will be displayed:

`I/O ERROR (IN XXXX)`

If an `ON ERROR GOTO` statement has been executed in the program mode, control is transferred to the specified error trap routine.

2. Abort processing

   Control jumps to label named `ABTDO` (address `A908`$_{16}$). The BASIC interpreter clears all variables, closes all files and initializes all I/O devices. Then, the following message is displayed:

   `ABORT (IN XXXX)`

18.2 Loading from expansion devices

The BASIC interpreter inhibits load from any device other than "`CAS0:`", "`CAS1:`", "`PAC0:`" and `COM0:`". Loading from any device other than these will result in an `FC` error. However, load from expansion devices can be enabled by rewriting the hook on the RAM (normally set to jump to the `FC` error routine). The RAM hook is 3 bytes long and has a format: `JMP XXXX`.

Write the entry point address of the program enabling loading from the expanded device into the address portion of the hook. For load processing, when control is returned from the `OPEN` routine, variable `ASCFLG` (one byte, address `068C`) is checked, and if `ASCFLG` is `00`, binary format load is performed.

The following two routes are used by the `OPEN` routine to set the value of variable `ASCFLG`.

1. `FF` is set in variable `ASCFLG` when the `A` option is specified in the `SAVE` statement and `00` is set when the `A` option is not specified. This data is written to the file header during program save and set in variable `ASCFLG` by the `OPEN` routine during load processing.

2. If the `A` option is specified in the `SAVE` statement, a value other than `FF` is written as the first character of the file. If the `A` option is not

specified, FF is written as the above character. Therefore, the value of ASCFLG can be set by reading of the first character of the file using the OPEN routine.

- Hook name: HKLOAD.
- Address: 05E2.
- Parameters:
  - (A): device number.
- Processing sequence:

```
HKLOAD  EQU   $05E2
FCERR   EQU   $8C70
LODCNT  EQU   $A6D0
; ....
        LDD   #LOADCK
        STD   HKLOAD+1
; ....
LOADCK  CMPA  XX        ; Check the device number
        BEQ   LOADOK
        JMP   FCERR     ; Give "FC Error"
LOADOK  JMP   LODCNT    ; Continue loading
; ....
```

18.3 ABORT processing

If an I/O operation is aborted by pressing the BREAK key, the BASIC interpreter initializes all devices and closes all files (communications channels). When one of the devices in the DCB table has been expanded, these devices will also have to be initialized if I/O to another device is aborted. This initialization is also performed using a hook.

- Hook name: HKABTD.
- Address: 063C.

Note: normally, 39_{16} (RTS command) is stored at address 063C.

18.4 RAM management

18.4.1 Application files

Application programs (BASIC interpreter, word processor, etc.) can use the RAM to store the data required by their systems as application files.

Application files are protected against use and accidental destruction by other application programs. Required data can be stored in these files in the same manner as data for BASIC programs can be stored in RAM files.

1. Before execution of an application program (Figure 18.2).

   All application files are stored in the upper addresses of the RAM.

2. During execution of an application program (Figure 18.3).

   The application program reserves a work area for itself by moving the application files stored at addresses lower than its own to addreses lower down in the free area. However, the location of this work area varies according to the status of the other application files. Therefore, if a fixed work area is required, the area immediately following the system area is reserved for this purpose. To secure work areas for execution, each application program expands its application files into the fixed and variable work areas.

3. Upon termination of application program execution.

   Upon termination of execution of an application program (power switch is turned OFF, RESET switch is pressed or normal completion), control returns to the Menu leaving the RAM allocation as it was during the execution of the application program.

   Then, when the same or another application program is selected from the Menu, the menu program calls the file reform routine for the files of the previously executed application program. The file reform routine selects only the required data from the fixed and variable work areas to create an application file and returns the RAM to the status in 1 above. Control is then transferred to the application program selected from the menu.

   For application programs which do not require application files, the free area is used as work area as shown in Figure 18.2. In this case, the file reform routine is not called.

18.4.2 RAM map

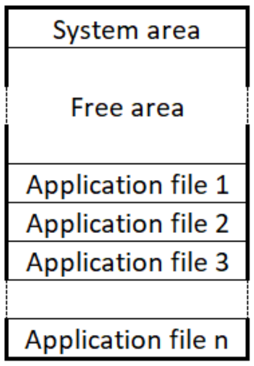


Figure 18.2: RAM map

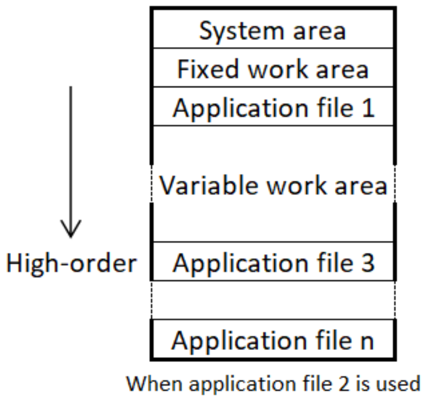


Figure 18.3: Ram map (2)

18.4.3 Data configuration

- `BASTAB`: indicates the beginning of the application file. When the system is initialized, the address set here is the same as that indicated by `RMLTAD`.

- `RMLTAD`: indicates the last address in the RAM+1. The value of `RMLTAD` is set when the RAM is checked during system initialization.

- `CNDADR`: indicates the entry point of the file reform routine. The address of the file reform routine for the application program is set in this variable when the application program is executed and the application files are expanded.

- `INITAB`: `INITAB` bit 6 is set (logic '`1`') to indicate that the files must be reformed before the next application program can be executed. This flag is set when the value of `CNDADR` is set.

  When this flag is set, the menu program calls the subroutine whose address is stored in `CNDADR` (file reform routine for the previously executed application program) before transferring control to the application program selected from the menu. This flag is reset within the subroutine after the application files are reformed. When application files are not used, this flag must not be set.

Figure 18.5 shows an example of when two application files exist simultaneously. The beginning of application file 1 is indicated by `BASTAB` while the end of application file 2 is indicated by `RMLTAD`.

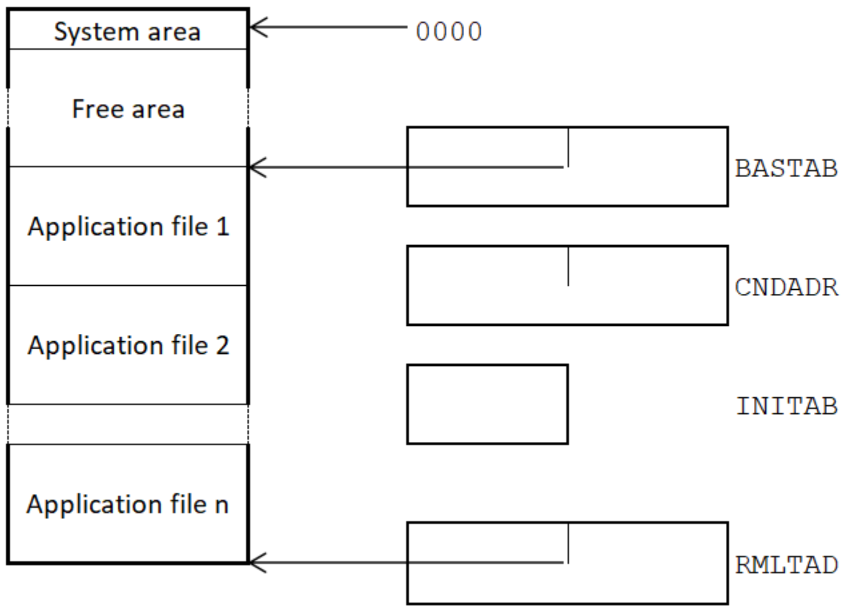


Figure 18.4: Pointers used for application files

1. File size

   The file size is shown by the first two bytes of the file in higher- and lower-order byte sequence. The starting address of the next application file can be obtained by adding the file size to the beginning address of the current file.

2. Application ID

   Application programs are assigned unique one-byte values which are used as IDs. These application IDs are used by application programs when searching for their files.

3. Data

   The data length is the file size−3 bytes. Data format is optional. Unique formats may be used for individual application programs.

18.4.4 Configuration of BASIC application files

BASIC application files must be stored at the end of the application file area.

1. Application ID

   BASIC: 80_{16}

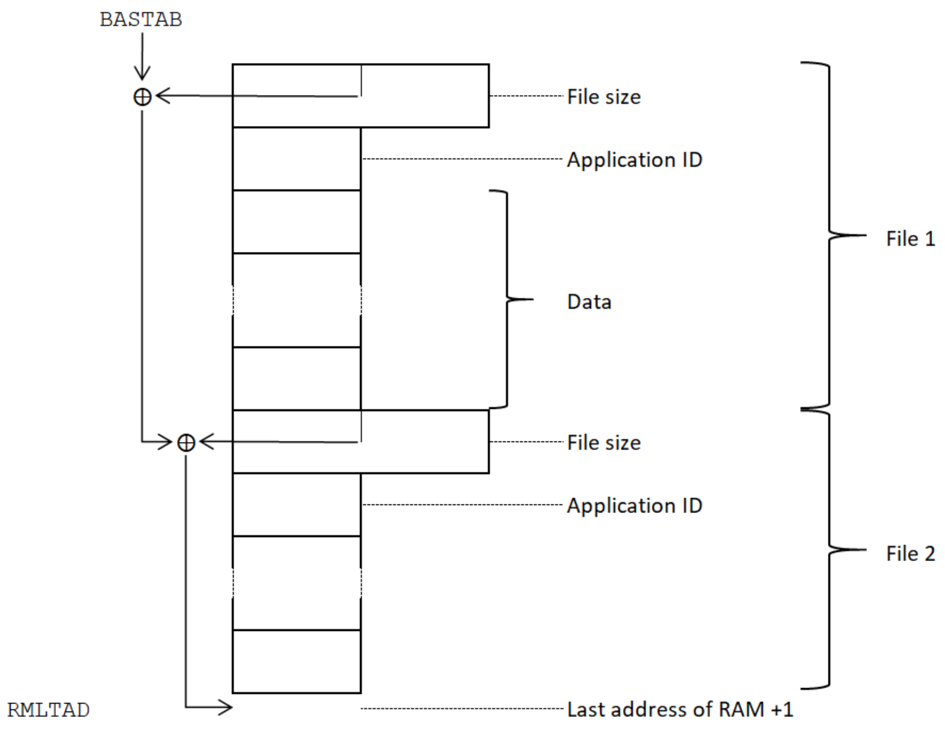


Figure 18.5: Use of pointers for two application files

2. Warm start hook

   The one- to three-byte machine language command stored in this hook is executed to execute BASIC warm start. $\mathtt{39}_{16}$ (`RTS` command) is set here when the system is initialized.

   When the expanded BASIC code is stored in the RAM, a `JMP` command (`C3XXXX`) is set in this hook to transfer control to the initialize routine for expanded BASIC.

3. Lowest address used by BASIC

   The address specified in the `MEMSET` statement is set.

18.5 Initializing extended BASIC

18.5.1 Expansion method

When executing warm start, the BASIC interpreter copies the DCBs and the DCB tables from the ROM and initialize the hooks and pointers. To expand

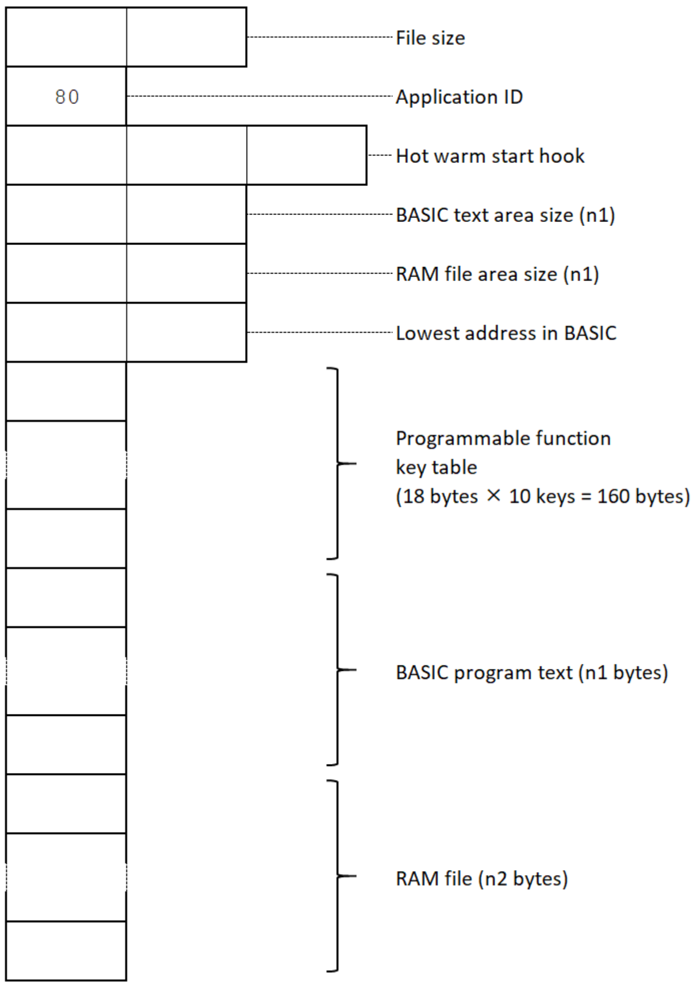

80
File size
Application ID
Hot warm start hook
BASIC text area size (n1)
RAM file area size (n1)
Lowest address in BASIC
Programmable function key table (18 bytes × 10 keys = 160 bytes)
BASIC program text (n1 bytes)
RAM file (n2 bytes)

Figure 18.6: Application file

BASIC, these hooks and DCBs must be changed after warm start has been executed. Three methods of expanding BASIC (ROM base, RAM base and disk base) are available.

After initialization has been completed, the BASIC interpreter executes BASIC expansion in the following sequence. The DCBs and hooks are rewritten by the initialize routines in ROM or RAM or by the `DISK` boot program.

1. Check executed for whether the expansion ROM has been set in the memory bank in which the BASIC interpreter is currently located. Control is transferred to 3. below, if the expansion ROM is not stored in this memory bank.

2. The initialize routine for the expansion ROM is executed.

3. Check executed for whether the floppy disk unit is available for serial communications. If the disk unit is not connected, control is transferred to 5. below.

4. The boot program is loaded from the floppy disk unit and then executed.

5. Warm start hook is executed (if RAM-base expansion is to be executed, a `JMP` command to transfer control to the initialize routine is set in this hook).

18.5.2 Expanded ROM format

Format for expanding BASIC on a ROM base is shown below.

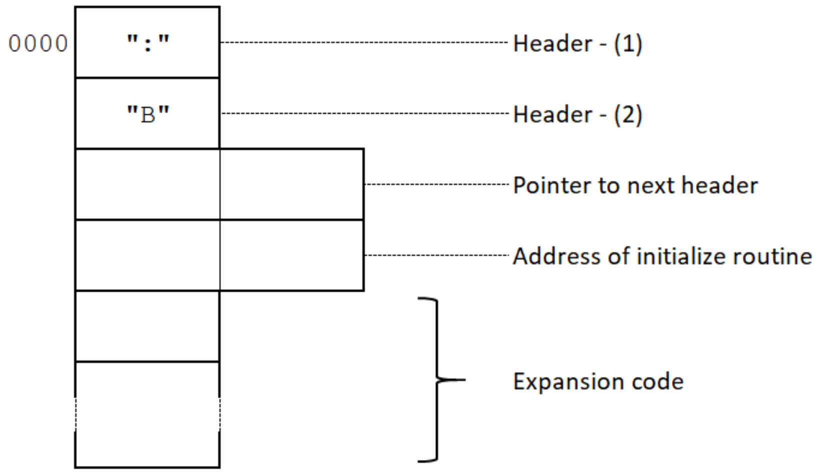


Figure 18.7: Expanded ROM format

Notes:

1. The expanded ROM for extended BASIC must be located in address 6000_{16}.

2. Other application programs may be stored in the same ROM with extended BASIC. However, the header of extended BASIC must be located at the starting address of the ROM.

18.5.3 Expansion on RAM base

Loading extended BASIC

The memory area for extended BASIC is reserved by creating a special application file at the end of the other application files. The procedure for loading extended BASIC is described below.

1. The BASIC interpreter is executed after initialization (`Ctrl`+`@`).
2. Load extended BASIC and the program to reserve the necessary memory area into the machine language area (`LOADM` command).
3. Execute the program for reserving the memory area.

   This program renews `BASTAB` and `RMLTAD` and reserves a RAM area sufficient to store extended BASIC. It then moves extended BASIC from the machine language area to these files. Also, the warm start hooks, etc., in the BASIC application file are rewritten and the initialize routine for extended BASIC is attached at the end of the initialize routine chain which starts from the warm start hook.
4. Transfer control to the BASIC interpreter warm start routine.

The above sequence makes extended BASIC resident in the RAM. Thereafter, when warm start is executed, the initialize routine in extended BASIC rewrites the DCBs and hooks to expand BASIC.

As the area reserved for extended BASIC is at the end of the application files area, it remains unaffected even if the application files are used by other application programs.

The extended BASIC codes must be assembled to enable their use at the destination addresses. However, these addresses of course vary with the current RAM capacity. In order to enable use of the codes irrespective of the RAM capacity, extended BASIC must be relocated after it is moved to the RAM.

Program for reserving extended BASIC area

The procedure for reserving the necessary memory area for extended BASIC is described below.

1. When control is transferred to the program for reserving memory area, the BASIC interpreter is already running and the BASIC application files are already extended. The file reform routine is therefore called to store only the necessary data in the application files (Figure 18.8).

```
LDX    CNDADR
JSR    0,X
AIM    #$BF,INITAB
```

2. Next, the BASIC application files are moved forward (`BASTAB` → `RMLTAD` −1) to reserve the area for extended BASIC (`BASTAB` is also updated).

   Simultaneously, (`RMLTAD`) is also updated and set at the head of the extended BASIC area to protect extended BASIC (Figure 18.9).

3. Extended BASIC, loaded simultaneously with the memory reserve program, is then moved to the newly reserved application files.

4. A jump command to transfer control to the initialize routine for extended BASIC is set in the warm start hook in the BASIC application file (currently, `RTS` command) or in the initialize hook for extended BASIC already existing in the RAM.

5. Control jumps to the BASIC interpreter warm start entry point.

```
LDX    $8004
JMP    0,X
```

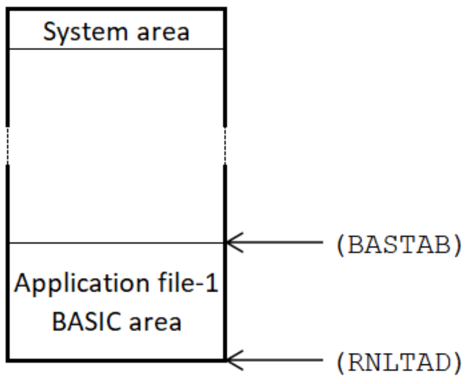


Figure 18.8: Before memory reserve

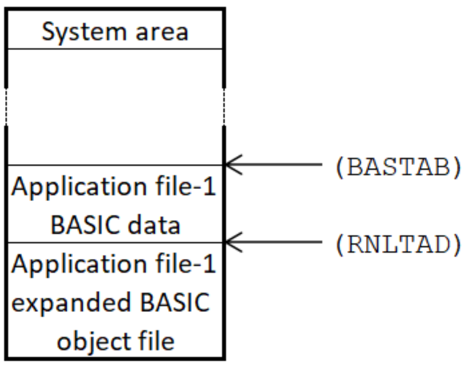
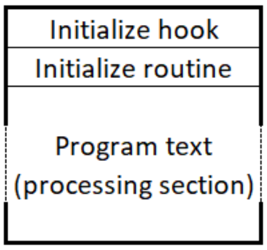


Figure 18.9: After memory reserve

Configuration of the extended BASIC object file

The configuration of the extended BASIC object file is shown below.



1. Initialize hooks

   The initialize hook consists of the 3 bytes shown below. When multiple extended BASICs reside in the RAM, the hook is used to link the different initialize routines.

   The initial value of the hook is `RTS` (39_{16}).

39	00	00

2. Initialize routine

   The initialize routine starts from the next address following the initialize hook. Each time BASIC is warm started, this routine rewrites the hooks, ADBs, etc.

   When the initialize routine is entered, the pointer to the sign-on message is stored in register (`X`). This is either the current BASIC sign-on message or else the sign-on message set by the previous initialize routine for extended BASIC. The pointer to the sign-on message must be set in register (`X`) when the initialize routine is exited. To display a sign-on message for extended BASIC, set the pointer for the sign-on message in register (`X`) on exit from the initialize routine for extended BASIC. The sign-on message will then be output when control is returned to the BASIC interpreter or when control is transferred to the next initialize routine for extended BASIC. If the set message is to be output when the initialize routine is entered, `STROUT` should be called on entry.

   If the above sign-on message is not to be output, the value of register (`X`) should be retained so that this register can be returned to its initial value on exit from the initialize routine. In this case, the normal message or the message set by the previous initialize routine will be output.

3. Chaining initialize routines

When multiple extended BASICs are to be expanded on the RAM, the initialize routines for all of those BASICs must be executed at warm start. First, as the warm start hook has been rewritten to transfer control to the first BASIC initialize routine, this routine is executed.

Upon completion of execution of the initialize routine, control jumps to the initialize hook. At this stage, if the initialize hook is still set to its initial value, the `RTS` command will be executed and control returned to the BASIC interpreter. If the initialize hook has been rewritten to jump to another BASIC initialize routine, that routine will be executed next. Initialize routines can in this way be chained and executed in succession until the `RTS` command is encountered.

Rewriting warm start and initialize hooks

The procedure for adding the initialize routine for an extended BASIC, newly loaded in the RAM, to the end of the execution chain starting from the warm start hook is described below.

1. The warm start hook in the BASIC application file is checked. If the value of the warm start hook has not been rewritten (that is, if it is still `RTS`), it is rewritten to jump to the initialize routine for extended BASIC.

   If the warm start hook has already been rewritten (if a jump command has been set), operation proceeds to 2. below.

2. The extended BASIC initialize hook at the jump destination of the warm start hook is checked. If it has not been rewritten, it is rewritten to jump to the initialize routine for the newly loaded extended BASIC.

   If the initialize hook has already been rewritten (if a jump command has been set), control is returned to 2. This operation is repeated until an initialize hook in which `RTS` has not been rewritten is encountered.

18.5.4 Extended BASIC work area

The following RAM area is used as the work area for extended BASIC irrespective of whether BASIC has been expanded on the ROM or on the RAM.

`0A38 - 0A3D` (6 bytes)

For RAM base expansion, if the work area is insufficient, a work area in the application files is reserved along with the area required for loading extended BASIC. For ROM base expansion, a RAM area is reserved with the application files as in RAM base. This area is then used as the work area (subroutines are set in the ROM and executed manually –`EXEC` command– after system initialize).

The same procedure is followed to retain data in extended BASIC.

18.6 System variables and hook table

18.6.1 System variables

1. `INITAB` (address `0078`$_{16}$, 1 byte)

   Bits 0 to 5 and bit 7 are initialize request flags. One bit is assigned for each application. The flag is set (logic "`1`") to indicate that initialization has been executed. It is reset at system initialize.

   The bit of this variable corresponding to the application program to be executed is checked prior to execution if the program requires initialization for its files, etc. If the flag is reset (logic "`0`"), initialize processing is performed to reserve the necessary work areas, etc., for the files and execution of the program is performed only after the `INITAB` flag becomes "`1`". If the flag is set (logic "`1`"), this means that the application program has already been initialized. It can therefore be executed immediately. `INITAB` flags are not assigned to application programs which do not require initialization.

   Bits currently are used as follows.

Bit 0	—	Menu program
Bit 7	—	BASIC interpreter

   Bit 6 is a file reform request flag. For application programs which require their files be expanded, the pointer to the file reform routine must be set in variable `CNDADR` and bit 6 of `INITAB` must also be set after file expansion has completed.

   The file reform routine is called by the menu program and resets bit 6 of `INITAB` after reforming the files.

2. `RMLTAD` (address `012C`$_{16}$, 2 bytes)

This is the pointer for the last address in the RAM +1. This variable is set at system initialize. Also functions as the pointer for the last address of the application files +1.

3. `BASTAB` (address `0134`$_{16}$, 2 bytes)

   Pointer to the starting address of the application files. Set to the same address as `RMLTAD` at system initialize.

4. `CNDADR` (address `0136`$_{16}$, 2 bytes)

   Pointer to the file reform routine. Set by the application program. Valid only if `INITAB` bit 6 is also set.

5. `DCTAB` (address `0657`$_{16}$)

   DCB table.

6. `DEVNUM` (address `063E`$_{16}$)

   Enables `LOAD` from expansion devices.

7. `ASCFLG` (address `068C`, 2 bytes)

   Specifies mode (ASCII or binary) for load. Set by the device `OPEN` routine.

   The BASIC interpreter interprets the flag status as follows:

   - `FF`: ASCII load.
   - `00`: binary load.

8. `OPTBUF` (address `068F`)

   The character string in the file descriptor used to specify options is set in this buffer. The option routine uses this data. The file descriptor option statement is set in this buffer in its original form. It is not placed in brackets. (`00`) is used as the end mark. If (`00`) is used as the first character, option is assumed not to have been specified.

18.6.2 Hook table

1. `HKLOAD` (address `05E2`$_{16}$)

   Enables `LOAD` from expansion devices.

2. `HKABTD` (address `063C`$_{16}$)

   Used to initialize expansion devices in case of `ABORT`.

18.6.3 Entry point table

	Label name	Address
(1)	ERROR	8433
(2)	ABTDO	A9D8
(3)	FCERR	8C70
(4)	LODCNT	A6D0

Appendix A

Serial communication protocol (EPSP)

A.1 Basic line specification

1. Transmission speed
2. Synchronization
3. Communication
4. Transmission
5. Response system
6. Error control
7. Transmission codes
8. Bit transmission sequence bit 0, bit 1,... bit 7

A.2 Transmission characters and sequence

`PS`	Request receiving side to prepare to receive data
`DID`	
`SID`	
`ENQ`	
`SOH`	Indicates start of header block
`STX`	Indicates start of text block
`ETX`	Indicates end of text block

ACK	Acknowledge
NAK	Negative acknowledge
DLE	Waits for WAK, acknowledge or transmission
ENQ	Prompt for block response
EOT	Releases data lines

PS must be ‘1’= 31_{16}. Control characters, DID and SID must be 8 bits (MSB=0).

A.3 Message format

A.3.1 Header format

SOH	Start of header
FMT	Text format 00: indicates that the master is transmitting a block 01: indicates that a slave is transmitting a block
DID	Destination ID
SID	Source ID
FNC	Text function
SIZ	Text size (in bytes) This value is the length of the text block (excluding STX, ETX and CKS) minus 1
HCS	Checksum of header block This is a value such that the lower 8 bits of the sum of SOH to HCS are 0

A.3.2 Text format

STX	Start of text
DB0	Data 0
DB1	Data 1
...	
DBn	Data n
ETX	End of text
CKS	Checksum of a text block The value of CKS is such that the lower 8 bits of the sum of STX to CKS are 0.

Text length excluding `STX`, `ETX` and `CKS` must be within 256 bytes.

A.4 Response to slave selection sequence

- `ACK` (acknowledgement)

  Indicates that the slave can receive a block. The master then initiates data transmission.

- `NAK` (negative acknowledgement)

  Indicates that the corresponding I/O device is not connected or that an error has occurred and the slave cannot receive data. The master then issues `EOT` and terminates the data link. The master will also send `EOT` to terminate the data link by transmitting if no response is received within a fixed period of time or an invalid response other than `ACK` and `NAK` is received after a selection sequence has been sent.

A.5 Header block transmission

A.5.1 Response to a header block

- `ACK` (acknowledge)

  Indicates that the slave has received a correct header block. The master proceeds to the next phase.

- `NAK` (negative acknowledge)

  Indicates that the slave has received an incorrect header block. In this case, the master repeats transmission of the same block. If the master still receives `NAK` after the block has been transmitted a specified number of times, it assumes a line error and terminates the data link (by send `EOT`).

- `WAK` (acknowledge and temporary wait)

  Indicates that the slave has received a correct block but that it cannot yet receive the next block. The master will wait and then issue `ENQ` to prompt a response from the slave.

- No response or invalid response

  If no response is made within a given time or a response other than `ACK`, `NAK` or `WAK` is received, the master will issue `ENQ` to prompt a response from the slave.

If no response is received even after `ENQ` has been transmitted a specified number of times, the master assumes an error and terminates the data link.

A.6 Termination

- When, after sending `ETX` to the slave, the master receives `ACK`, it sends `EOT` to the slave and terminates the data link.
- When a transmission error occurs after the data link has been established, or during data transmission, the master will terminate the data link by transmitting `EOT`.

A.7 Time supervision

1. Number of selection sequences transmitted

   The master will repeat the selection sequence after receiving a response other than `ACK` from the slave for the number of times listed in the table below.

	Mode 0	Mode 1
`NAK`	One time	One time
No response or invalid response	Three times (at 1s intervals)	Three times (at 3s intervals)

2. Number of transmitted `ENQ`s (response retransmit request)

	Mode 0	Mode 1
No response or invalid response	Three times (at 1s intervals)	Three times (at 3s intervals)

3. Timers

	Mode 0	Mode 1
Response wait timer	1s	3s
Interblock supervision	32s	96s
Character supervision	1s	3s

A.8 Terminal numbers

- `31`$_{16}$: Floppy disk drive A

- $\texttt{32}_{16}$: Floppy disk drive B
- $\texttt{33}_{16}$: Floppy disk drive C
- $\texttt{34}_{16}$: Floppy disk drive D

Terminal numbers (slave)	:	$\texttt{30}_{16}$ through $\texttt{3F}_{16}$ for mode 0
		$\texttt{40}_{16}$ through $\texttt{5F}_{16}$ for mode 1
Center number (master)	:	$\texttt{20}_{16}$

A.9 Omission of a header block

If the terminal previously transmitted to is still selected and the header to be transmitted is the same as the last transmission, the header may be omitted. In this case, the master need only transmit the data block following `STX`. The slave treats this data block without header as if it included the header of the previously received data block.

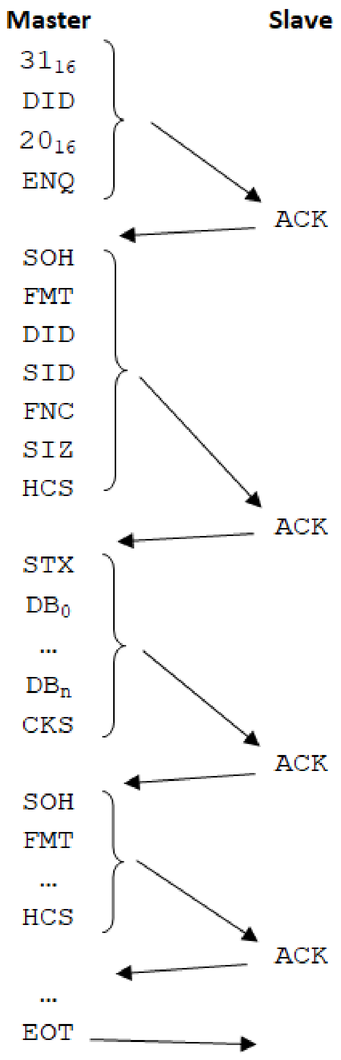
A.10 Transmission conditions supported by ver.1

Transmission speed is 38.4Kbps. Mode 0 is used for time supervision. Header block cannot be omitted.

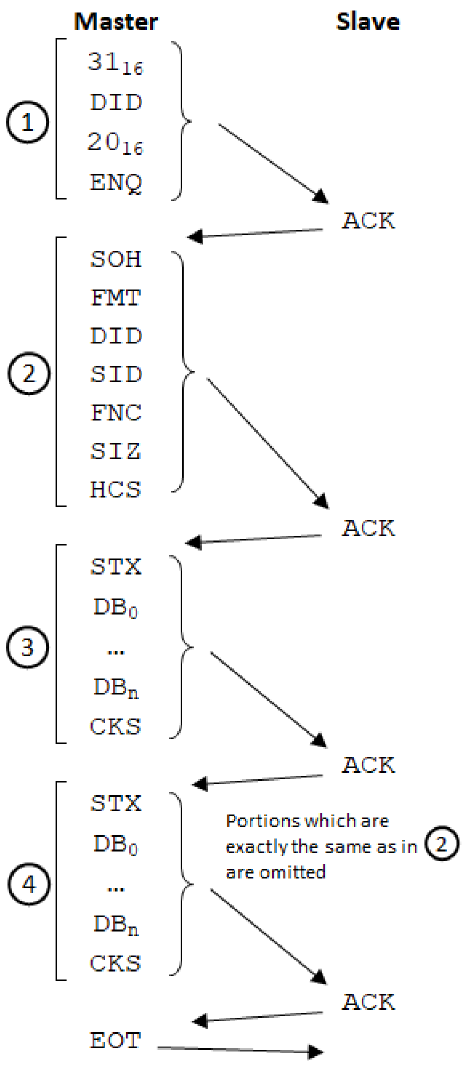
A.11 Transmission procedure diagrams

A.11.1 Without errors

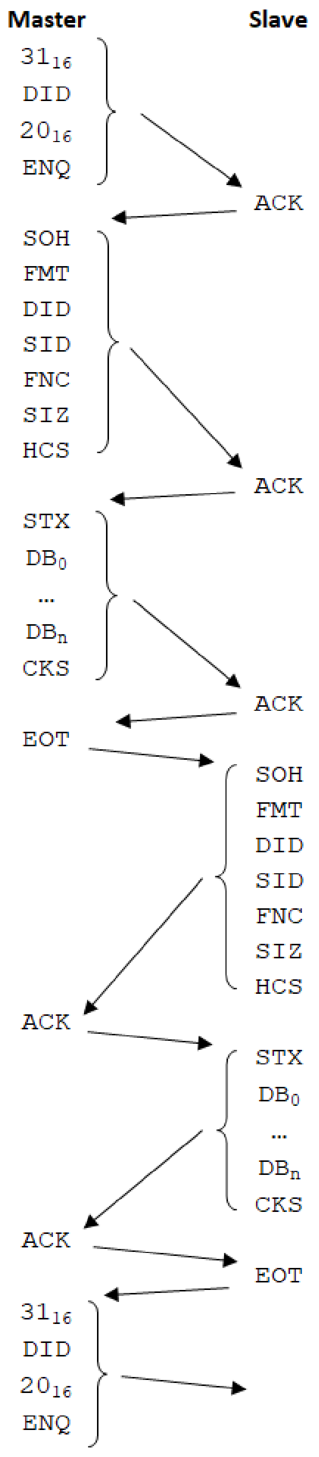
1. When the slave does not send a data block to the master in response to the master's transmission and a header is not omitted.



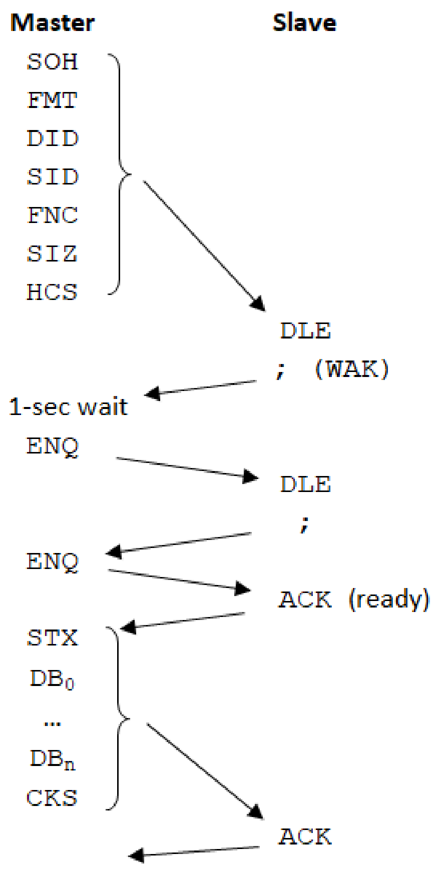
2. When the terminal does not transmit a data block to the master in response to the master's transmission but the header is omitted.



3. When the terminal transmits a data block to the master in response to EOT from the master and the header is not omitted.

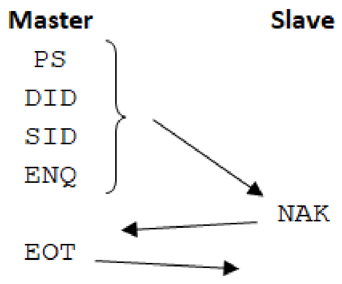


4. When the slave responds with `WAK` to a block transmission with header from the master.

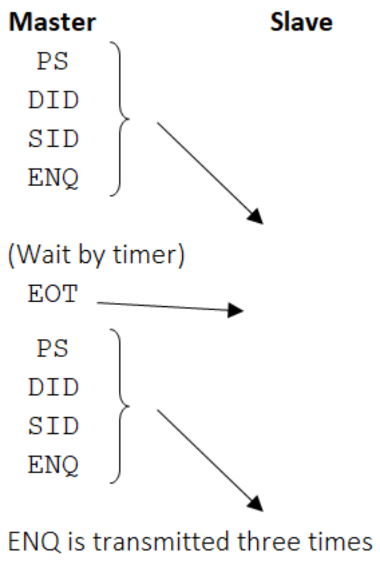


A.11.2 With errors

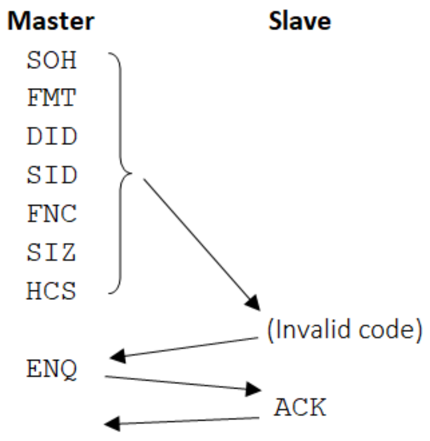
1. When the slave responds by sending `NAK` in response to `ENQ` from the master.



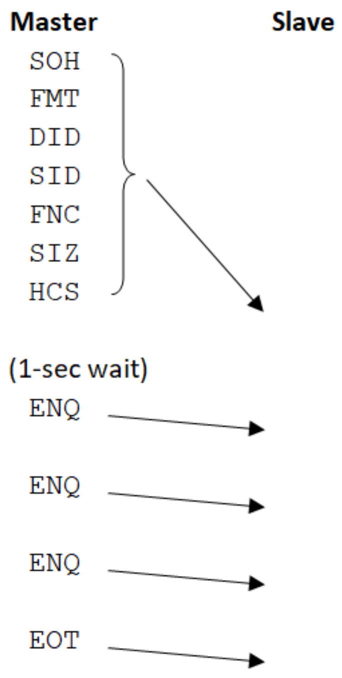
2. When no response or an invalid response is received from the slave in response to `ENQ` from the master.



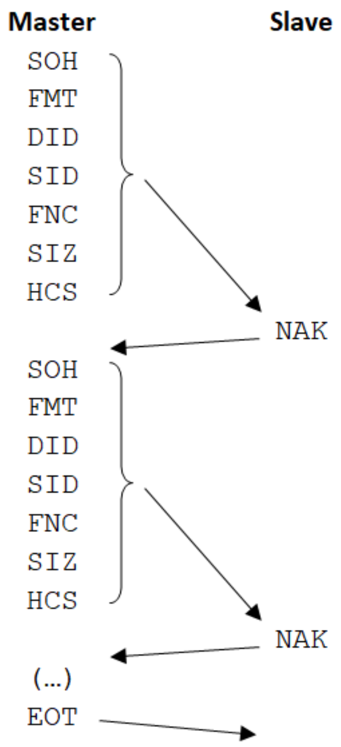
3. When the master receives an invalid code from the slave in response to transmitting a header to the terminal.



4. When no response is received from the slave in response transmission of a header from the master.



5. When NAK is sent from the slave in response to transmission of a header from the master.



If the master receives `NAK` three or more times, it terminates the data link by transmitting `EOT`.

If the slave transmits `NAK` three times in succession, the master will not send the header but will send `EOT` to terminate the data link.

6. When the master receives an invalid response code from the slave in response to text transmission, the master handles this as in 3 above.

7. If no response is received from the slave, the master handles this as in 4 above.

8. If the slave responds with `NAK` when the master transmits text, the master handles this as in 5 above (text retransmission).

9. When the master does not receive a correct response after sending `EOT` to the slave.

   (a) If there is no response, the master waits for 1 second and terminates the data link by sending `EOT`.

   (b) If the master receives a code other than `EOT` from the slave, it terminates the data link by sending `EOT`.

10. When the slave has not correctly received and responded to `EOT` sent from the master (when response from the slave is necessary).
    (a) If the slave has not received `EOT` and does not respond to the master, the master will wait (3 seconds in mode 0), terminate the data link and restart the link procedure from the beginning.
    (b) If the slave receives a code other than `EOT`, the master assumes that the terminal has made no response. If the slave does not receive `EOT` after the master has sent `EOT` the specified number of times (3 times), the center returns to the link start procedure.
11. If the slave does not transmit a header after the master transmits `EOT`, the master requests the slave to transmit the header by retransmitting `EOT` after waiting for a given time. If the slave does not transmit the header even after `EOT` has been transmitted the specified number of times, the master assumes that an error occurred and terminates the data link.

Function character code table

	0	1
0		DLE
1	SOH	
2	STX	
3	ETX	
4	EOT	
5	ENQ	NAK
6	ACK	
7		
8		
9		
A		
B		
C		
D		
E		
F		