Information Processing in Japan Vol. 9, 1969

TSS I/O Control Program

EIITI WADA*

1. *Introduction*

Remote terminal control is one of the important problems of the time-sharing systems (TSS). In one system, the central processing unit (CPU) controls every character of every terminal, character by character, but in another system, special purpose machine is connected, which controls terminals and exchanges information with the CPU in the unit of one line of message. The first one increases the overhead to the CPU, while the latter decreases the flexibility.

When the experiment of the TSS at the Computer Centre of the University of Tokyo using the HITAC 5020 was proposed, this problem had to be solved at the very beginning. It was decided that the special transmission and reception controller (TRC) which shared the high speed memory which the CPU and would operate according to the stored program should be constructed and installed and should control the remote terminals. Two of the TRCs had already been constructed and installed both at the Computer Centre of the University of Tokyo and the Central Research Laboratory of Hitachi Ltd.

This system configuration forced us to prepare the programs for the TRC as well as the programs for the CPU (TSS monitor) and to design the control method and the information transfer rule between the TRC program and the TSS monitor. So far the TRC program has been completed. In the following chapters, a brief explanation of the TRC, relations between the TSS monitor and the TRC program, precise structures of the TRC program, and main features of our TSS are described.

2. *Rough Illustration of the TRC*

Although the cycle time of the high speed core memory of the HITAC 5020 is 2 μsec, the core memory is used on 4 μsec interval. It is because the transmission between the memory register and the storage device is parallel in 2 μsec, but the transmission between the memory register and the CPU is made serially, and for this serial transmission, one memory cycle of 2 μsec is consumed. Accordingly, the storage device is read or written in every other memory cycle. The proposal for the TRC was that, another new memory register should be prepared for the TRC and the TRC should use the same storage device in the

---

This paper first appeared in Japanese in Joho Shori (the Journal of the Information Processing Society of Japan), Vol. 9, No. 6 (1968), pp. 335~343.

* Faculty of Engineering, University of Tokyo.

idle cycle time of the CPU, thus the memory reference speed by the CPU would not be decreased and at the same time the TRC could use as much storage locations as it wanted, and moreover, there would be no necessity for information transmission between two processors. Instruction format of the TRC was designed to be the same as that of the CPU, and registers and instruction repertories of the TRC were almost the subset of those of the CPU. Actually, some special instructions for the TRC were designed and some of the registers were assigned to have special functions. Therefore the TRC was not strictly the subset of the CPU, but still the TRC program could be written in an assembler language for the CPU, two programs were linked by the function of the assembler and the loader program, and some routines which were coded to use only the common instructions and registers could be executed on either processor.

The TRC has its own memory protection, real time clock, indicator register, interruption facility, priority and normal mode flipflop. Locations 0 to 31 of the memory of the HITAC 5020 are at the same time registers of the CPU composed of the delay lines. Locations 32 and upward are the high speed core memory which are shared with the TRC. Locations less than 32 referred to by the TRC are also the delay line registers exclusively prepared for the TRC (Figure 1). The real time clock, indicator register, input/output registers are in this dealy line part. Interruption addresses for the TRC were assigned to locations 256 and 257.

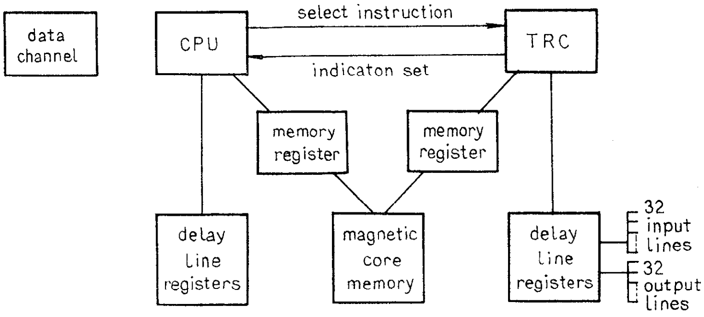


Fig. 1.

The TRC has the function to communicate with the 32 (or 64) teletypewriters at the remote terminals. In the basic conception of the proposal, the communication mechanism was very simple. Every bit of the 32 bits input register was tured to 0 or to 1, reflecting the current direction of the input line. The TRC program should sample each bit to decipher the input character. Each output line sent out the current according to the corresponding bit of the 32 bits output register. Since the system is very simple, other types of remote devices can be connected to the TRC, and moreover, one terminal can use plural number of lines. The above type of line control by the TRC is called the software mode.

The software mode is extrewely flexible, but for some purposes its control speed is too slow. It is desirable to sample more than 10 times per bit for the input signal. For the 32 input lines at the 200 bits/sec speed, sampling interval should be 15 μsec, which is almost impossible for the TRC. Therefore, later on, the special circuit was built to handle the teletypewriter input/output lines. This type of line control is called the hardware mode.

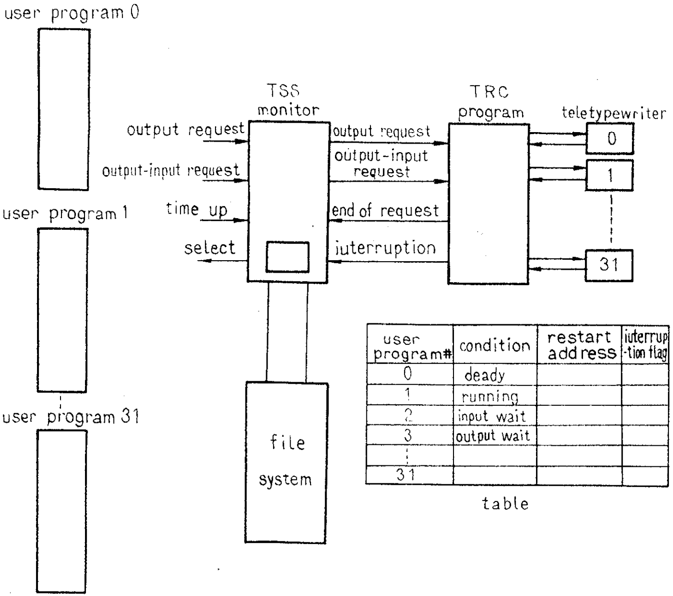
In the hardware mode, number of bits per second, number of bits per character must be in a specified location for each line group. A line group consists of 16 lines. Then input lines are monitored by the circuit and when one character is received from any line, this character code and the completion flag are stored in a predetermined location to the line and an interruption to the TRC occurs. For the output, when a code pattern and a flag are placed in another predetermined location to the line, then the code is sent out at a specified speed. At the completion of one character output, the interruption takes place.

Besides the above input/output completion, real time clock and calling from the CPU interrupt the TRC. In order to call the TRC, the CPU has to issue the select instruction which causes the interruption to the TRC. On the other hand, the TRC can call the CPU by turning on a special bit in the indicator register, which is subsequently cleared by another select instruction from the CPU.

3. *The TSS Monitor and the TRC Program*

3.1 *Control Locus of the CPU and the Standard Operation of the TRC*

Figure 2 shows the relations of the TSS monitor, the TRC program and the



user program#	condition	restart address	iuterrup-tion flag
0	deady		
1	running		
2	input wait		
3	output wait		
⋮			
31			

Fig. 2.

time-shared user programs. Normally one of the 32 user programs is running. In the meantime, the said user program may request output, or output-input, or end of time slice may happen. Then the TSS monitor receives the control, sets the condition of the user program in a table (output wait, input wait or ready) and swaps out the program. Then next, as for the requests, the monitor relays them to the TRC. Now control is relinquished to the next user program.

On being requested, the TRC program begins the interaction with the remote terminal. At the end of this interaction, it calls back the TSS monitor with the end condition parameter expecting the modification in the user condition table.

3.2 *Interruption to User Program*

In our system, the teletypewriter is always ready for typing in. However, typing in at the non-input mode is interpreted as an interruption signal to the user program by the user, and control in a user program jumps to the interruption service routine. Depending on the typed-in code, this routine may work in various ways, i.e., it stores the characters during the execution of the program without waiting for the input, or if the character is decoded as a quit signal, this routine halts the program. The TRC program alerts the CPU with the received code and the notice that it has received an interruption, when it receives a quit code at the input mode, or any code at the non-input mode. This information will cause the execution of interruption service routine in the user program when it is selected next.

3.3 *Connection of Remote Terminals and the TRC Program*

At present, two teletypewriters of 110 bits/sec speed are connected with full duplexed lines to the TRC and they are serviced in the hardware mode. (The experiment in the software mode to control other types of input/output devices than the standard teletypewiters is scheduled to be perfomed in the later stages.) Since the transmissions in both directions are independent, aforementioned quit and interruption system was chosen. Instead of the direct printing, typed-in characters are sent to the TRC once, and then echoed back to the teletypewriter to be printed. The time delay between typing and printing is negligibly small at the 110 bits/sec line. Input character which causes the interruption or quit is not sent back because if it is typed while the TRC is sending the output message, the sent back character would damage the output information. However, to make sure that the character is received, the bell signal is sent back which may well be inserted among the output information.

3.4 *Output-Input Request*

Interruption or quit facility by typing in at the non-input mode makes it necessary to indicate the time of the input mode. For this purpose, two standard rules which indicate the beginning time of the input mode are settled. The first rule is star symbol or currency symbol at the third printing position and the teletypewriter stop. The second is the three continuous periods and the teletypewriter stop. The first rule is for the input at the new line of normal or command levels, while the second is for answering to the question preceding the three periods. Thus the output always exists before the input. This output message is connected in the standard portion of the user program, and sent out through the TSS monitor. Here, if the output request is separated from the input request, two difficulties arise.

The output request swaps out the user program. After the output, this user program will be selected and excuted only to request input and be swapped out again. Secondly, there exists some delay between the rule of input indication and the beginning of the actual input mode, which delay might modify the input character into the interruption symbol. Considering these conditions, the input request is paired with the output request.

3.5 *System Freezing*

In order to make it possible for the TSS to freeze the operation, the TRC program must not have the input editing facility. The input editing should be done by the user program. When the system freezing is requested, the TSS monitor holds all the subsequent requests to the TRC. At the time when every request issued in advance to the system freezing command completes, it is ready to freeze. The output requests complete automatically, but the input requests depend on the user. In our system, the input is requested with time allowance, and even though the user does not type in the terminating symbol, input could be completed according to the allowance. Moreover it is hoped that the time allowance should be reset if one line cancelling is commanded. But, if the TRC program does cancelling action and resets the time allowance without informing the TSS monitor, and if the cancelling is command repeatedly, the TSS can not be freezed forever.

3.6 *Calling from the CPU to the TRC*

The TRC call is executed as follows. 1 word RQCPU, 1 word RQCPU1, 1 word RQCPU2, and 16 words OUTBUFFER are prepared for each user as shown in Figure 3, in the location into which the CPU can write, but the TRC cannot (CPU area). The output information is loaded in the OUTBUFFER. RQCPU1 which consits of 3 areas of 8, 8, 16 bits, is loaded with output character count, input character count limit, and input time allowance, RQCPU2 of four 8 bits areas is loaded with four terminating codes. Finally, RQCPU is loaded with the request code. MCP (master control program) call executes the interruption into the TRC.

Of these communication channels, since the user programs are being swapped out during the input/output operation, RQCPU1 and 2 would not be modified while the TRC is servicing. As for the RQCPU, however, the TRC searches

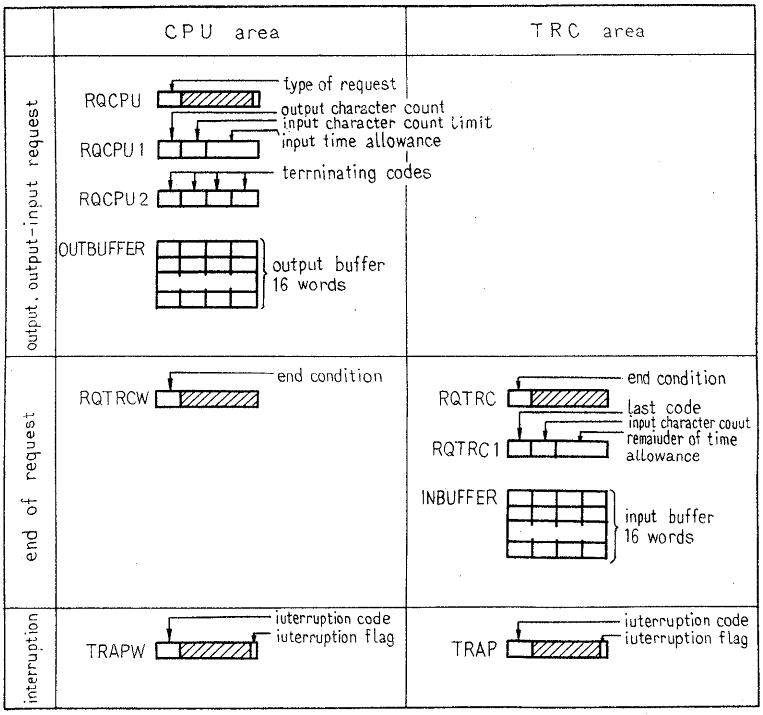


Fig. 3.

the line for which RQCPU≠0, and resets it to zero. Although the information necessary for the RQCPU is a few bits, one whole 32 bits word is assigned to each RQCPU, since if, say, 4 RQCPUs are packed into a word, and setting by the CPU and resetting by the TRC take place nearly simultaneously, setting and resetting might accidentally fail.

3.7 *Calling from the TRC to the CPU*

Two kinds of calls were designed for the TRC, i.e., end of request call and interruption call. For the end of request call each line possesses 1 word RQTRC, 1 word RQTRC1, 16 words INBUFFER in the TRC area, and 1 word RQTRCW in the CPU area. Input information is placed in the INBUFFER, the last received code, number of input characters, remainder of time allowance are stored in the RQTRC1 in the corresponding pattern to the RQCPU, and the end condition is inserted in the left 8 bits of the RQTRC. The end condition 1, 2 or 3 indicate the end of request by a terminating code, by time over, or by character count. At the end of the output request, end condition is set to 4. After the above informations are stored, the TRC turns a bit in the indicator which is connected to the CPU to 1. Because this bit can be reset by the CPU using the select instruction, the same accident shown in 3.6 seems to occur. However as the indicator is one of the arithmetic registers, the TRC can set the bit to 1 by one action, e.g., by addition, and exclude the critical racing problem.

On receiving the interruption from the TRC, the CPU in a priority mode copies the RQTRC to the RQTRCW and clears the RQTRC to 0. The TSS monitor recognizes the request from the TRC through the RQTRCW, which can be reset by the TSS monitor (running in a normal mode).

When the TRC is interrupted by the teletypewriter, it puts the flag bit and the received code in the one word location TRAP for the line, then interrupts the CPU. The TRAP is also copied to the TRAPW and cleard by the CPU.

The TRC puts the information into the RQTRC and TRAP, and never reads out. However it reads out the TRAPW to know whether the former quit code is left there unprocessed. The TRC reads the TRAPW when it proceeds from output to input processing the output-input request, and if quit is still there, the TRC suppresses the input action. The reason of this decision is as below. Suppose a user wanted to stop his program by pushing the quit key (one of the function keys is assigned as a quit key). The quit code is then copied to the TRAPW and waits there until his program is selected and executed next. Suppose again that the user program is being swapped out during the output-input request and when the quit is sent, the teletypewriter is outputting. The quit is then desired to become effective as soon as the output finishes without troubling the user to send the terminating code. In order to have it so, the TRC has to look at the TRAPW. The end condition is of the output request.

4. *The TRC Program*

4.1 *Interrupt Processing*

The general flow chart of the TRC program is shown in Figure 4. On being interrupted, the TRC program checks the reason and line number, and starts an appropriate process in a normal mode to detect memory protect violation, suppressing further interruption by masks. When the process comes to an end, it goes on to process the next interruption if some are waiting. Otherwise, it halts.

Reasons of the interruptions are

1. initialization by the CPU,
2. output request by the CPU,
3. output-input request by the CPU,
4. completion of one character output,
5. completion of one character input, and
6. time up of the real time clock.

While the character counters and storages for the various flags are prepared one set for each line, the TRC program has only one time management throughout the whole system, because it has to manipulate the list-like data structure of time information.

Output request makes the TRC program set the character count, and if

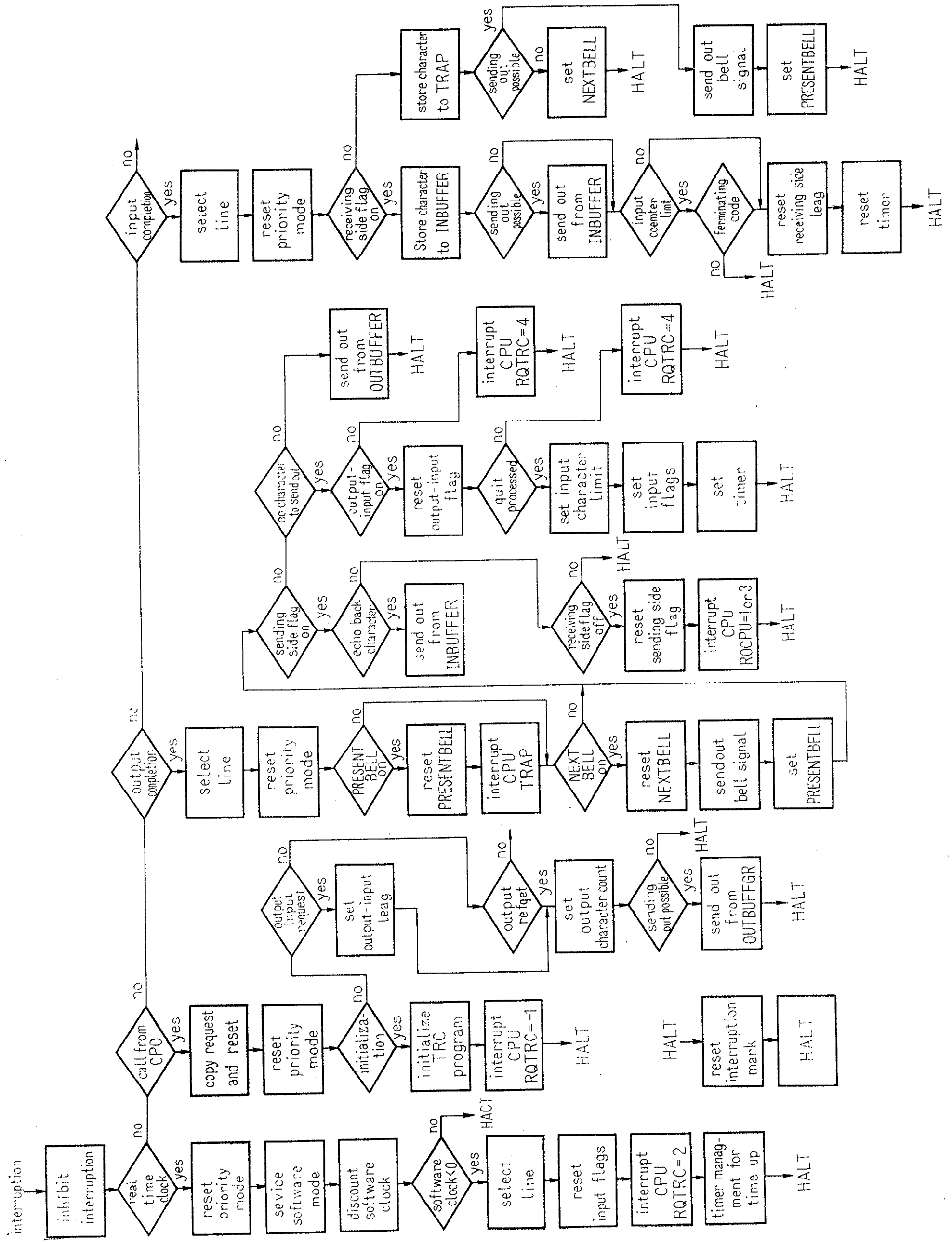

interruption
inhibit interruption
real time clock
yes
no
reset priority mode
sevice software mode
discount software clock
software clock<0
no
HALT
yes
select line
reset input flags
interrupt CPU RQTRC=2
timer management for time up
HALT
call from CPO
yes
no
copy request and reset
reset priority mode
initialization
yes
no
initialize TRC program
interrupt CPU RQTRC=-1
HALT
output input request
yes
no
set output-input leag
output refget
yes
no
set output character count
sencing out possible
yes
no
HALT
send out from OUTBUFFGR
HALT
HALT
reset interruption mark
HALT
output completion
yes
no
select line
reset priority mode
PRESENT BELL on
yes
no
reset PRESENTBELL
interrupt CPU TRAP
NEXT BELL on
yes
no
reset NEXTBELL
sendout bell signal
set PRESENTBELL
sending side flag on
yes
no
echo back character
yes
no
send out from INBUFFER
receiving sideflag off
yes
no
HALT
reset sending side flag
interrupt CPU ROCPU=1or3
HALT
no character to send out
yes
no
send out from OUTBUFFER
HALT
output-input flag on
yes
no
interrupt CPU RQTRC=4
HALT
reset output-input flag
quit processed
yes
no
interrupt CPU RQTRC=4
HALT
set input character limit
set input flags
set timer
HALT
input completion
yes
no
select line
reset priority mode
receiving side flag on
yes
no
Store character to INBUFFER
sending out possible
yes
no
send out from INBUFFER
input coemter limt
yes
no
terminating code
no
HALT
reset receiving side leag
reset timer
HALT
Store character to TRAP
sending out possible
yes
no
set NEXTBELL
HALT
send out bell signal
set PRESENTBELL
HALT

Fig. 4.

sending out is possible, the TRC sends out the first byte. After which it continues sending out when the output completion of the line occurs until the whole characters are sent out. On sending out the requested number of characters, it interrupts into the CPU informing the end of request. In case of the output-input request, it sets the output-input flag and begins the output. At the end of the output process, it checks the flag and the waiting quit in the TRAPW, then proceeds to input. Two input flags, namely receiving side flag and sending side flag, and timer are set. At the input completion while the receiving side flag is on, the received character is stored in the INBUFFER. If sending out it possible, it sends back the echo signal. Input completion or output completion increases or decreases the diflerence counter respectively. Diflerence 0 means that sending out is possible. Furthermore input completion process examines the character count and termination codes and if it finds out the end of input record, the receiving side flag is reset. On the output completion, if the difference counter is 0 and receiving side flag is off, then sending side flag is reset and end of request is returned to the CPU.

Input completion while the receiving side flag is off becomes the interruption signal. If sending out is possible, bell signal is sent out and the PRESENTBELL flag is set, if not, the NEXTBELL flag is set. Output completion has to do two more things. If the PRESENTBELL flag is on, it sends the interruption to the CPU. If the NEXTBELL flag is on, resetting the flag it makes the PRESENTBELL flag on and sends out the bell signal.

4. 2 *Timer Management*

The timer management routine has two entry points, i.e., registration and cancellation. Input parameters for the registration are line number and the time allowance, while for the latter only the line number is parametered. The time record is stored in the order of urgency with the line number and the time difference from the preceding item. The first one is set on the real time clock and counting down is carried on.

The registration follows the next steps.

(1) If the timer interruption inhibit bit is on, make the bit off and start the counting down in the real time clock.

(2) If inhibit bit is off, compare with the remaining time in the clock and decide which is more urgent. If the new registration is more urgent, set this time in the clock and put the record of the remaining time at the head of the list.

(3) If the new registration is less urgent, trace the list up to the proper place and insert the new item there.

In case of cancellation, the next steps must be followed.

(1) If the line number corresponds to that of the present counting down, add

the first item to the clock.

(2) If the list has no item, set the inhibit bit on.

(3) If the line number is not that in the clock, trace the list and remove the item by adding the time to the succeeding item.

Now, in the course of the counting down, if the real time clock becomes zero, the TRC is interrupted, and some output-input request is terminated by time over.

This invokes automatically the cancellation for the terminated line. Besides, the TRC program has the function to measure the servicing and idling time for the statistical purposes.

4.3 *Programming Language*

As expected in an earlier stage, for the TRC programming, the assembler language was found to be very useful except that since the absolute address coding was impossible in the HITAC 5020 assembler some tricky coding had to be used for some special locations for the TRC hardware.

5. *Acknowledgment*

The author would like to thank Professor H. Takahasi, Professor S. Moriguti, Professor T. Motooka and Professer E. Goto for the suggestions and ideas offered by them as well as Mr. H. Kameda, Mr. K. Sakuma and Mr. Y. Sakamoto for their contributions to this work. The author also would like to express his sincere thanks to the members of the Computer Centre of the University of Tokyo and the TSS group of the Central Research Laboratory of Hitachi Ltd., for their cooperation and valuable discussions held with them.