SYSTEM DEVELOPMENT CORPORATION

THE ARPA-RDC-T/MRDC COMPUTER LABORATORY

FINAL REPORT ON
CONTRACT DAAH01-69-C-1812

RICHARD G. BEELER,
Project Leader

30 JUNE 1971

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) System Development Corporation 2500 Colorado Avenue, Santa Monica, Calif. 90406	2a. REPORT SECURITY CLASSIFICATION 2b. GROUP

3. REPORT TITLE
THE ARPA-RDC-T/MRDC COMPUTER LABORATORY. FINAL REPORT ON CONTRACT DAAH01-69-C-1812

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Final Technical, June 1969 - June 1971

5. AUTHOR(S) (First name, middle initial, last name)
Richard G. Beeler

6. REPORT DATE 30 June 1971	7a. TOTAL NO. OF PAGES v, 96	7b. NO. OF REFS 0

8a. CONTRACT OR GRANT NO. DAAH01-69-C-1812 b. PROJECT NO. None c. ARPA Order No. 1427 d.	9a. ORIGINATOR'S REPORT NUMBER(S) TM-4635/000/01 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT
This document may be further distributed by any holder only with specific prior approval of Director, Advanced Research Projects Agency, Washington, D.C. 20301.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY ARPA Research and Development Center - Thailand

13. ABSTRACT

The principal focus of the project has been the production of a time-sharing system to demonstrate the potential of interactive computing in the Thai environment. In addition to the production of the resource-sharing system, activities included the development of a system for Thai transliteration, operation of the computer and its associated "service bureau," production of utility programs, and production and maintenance of hardware and software for attaching the various consoles. The system of Thai transliteration was developed to determine the extent to which translitera-tion can be performed by a computer, given only a Thai word as input; the work was divided into orthographic research and computer programming. Problems of maintaining the computer, training operators, and handling documents were solved by methods standard in the United States, although the Thai environment required more personal attention to users' problems and considerably more time in teaching.

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT

TM(L)-4635/000/01

SYSTEM DEVELOPMENT CORPORATION

THE ARPA-RDC-T/MRDC COMPUTER LABORATORY

FINAL REPORT ON
CONTRACT DAAH01-69-C-1812

RICHARD G. BEELER,
Project Leader

30 JUNE 1971

THIS RESEARCH WAS SPONSORED BY THE ADVANCED RESEARCH PROJECTS AGENCY (ARPA) OF THE DEPARTMENT OF DEFENSE UNDER ARPA ORDER 1427 AND WAS MONITORED BY ARPA AND BY THE U.S. ARMY MISSILE COMMAND (USAMICOM), REDSTONE ARSENAL, ALABAMA, UNDER CONTRACT NUMBER DAAH01-69-C-1812. VIEWS AND CONCLUSIONS EXPRESSED HEREIN ARE THE PRIMARY RESPONSIBILITY OF THE AUTHOR OR THE CONTRACTOR, AND SHOULD NOT BE INTERPRETED AS REPRESENTING THE OFFICIAL OPINION OR POLICY OF USAMICOM, ARPA, DOD, OR ANY OTHER AGENCY OF THE GOVERNMENT.

THIS [redacted] OF
DIR [redacted]

REPORT SUMMARY

TITLE:	The ARPA-RADC-T/MRDC Computer Laboratory		
AUTHOR:	Richard G. Beeler		
CONTRACTOR:	System Development Corporation		
CONTRACT:	DAAH01-C-1812, ARPA Order 1427		
CLASSIFICATION:	Unclassified	DATE:	30 June 1971

The principal focus of the project has been the production of a time-sharing system to demonstrate the potential of interactive computing in the Thai environment. The system that SDC has produced is one of the most complex resource-sharing systems ever worked on in Thailand. The system must be considered incomplete, however, until certain capabilities, such as sophisticated debugging techniques, can be implemented.

In addition to the production of the resource-sharing system, activities included the development of a system for Thai transliteration, operation of the computer and its associated "service bureau," production of utility programs, and production and maintenance of hardware and software for attaching the various consoles. Related planning and on-the-job training were included in all of these activities. A list of tasks and subtasks and the manner in which each was successfully concluded is given in Section 3; the work is discussed in greater depth in Section 4.

The system of Thai transliteration was developed to determine the extent to which transliteration can be performed by a computer, given only a Thai word as input. The work was divided into orthographic research and computer programming. As an articulation between the two, a string-processing language, TRANS, was devised. All important functions are handled by the TRANS rules. Words or syllables that are exceptions to these rules are dealt with by a separate program in which each is replaced by a new string of characters that enables the TRANS rules to operate. The system now contains 175 TRANS rules and 1,800 exception sequences. Test results of 96% correct were achieved by the end of the contract period. Section 4.3 contains a nontechnical description of the work accomplished in this area; Appendix B provides a technical discussion of the TRANS language; Appendix C provides a technical discussion of TRACTS (Thai/Roman Computerized Transliteration System).

* The authors wish to acknowledge the kind cooperation and assistance of the Military Research and Development Center, Thailand in the performance of all the work described herein.

Problems of maintaining the computer, training operators, and handling documents were solved by methods standard in the United States, although the Thai environment required more personal attention to users' problems and considerably more time in teaching. Equipment installed in the Laboratory is listed in Section 2. Problems overcome in its production, implementation, and support are reviewed in Section 4.

Section 5 describes contract activities from October 30, 1970 to June 30, 1971.

TABLE OF CONTENTS

1. INTRODUCTION

This report records the activities of the System Development Corporation in bringing to a successful conclusion the tasks required of it under Contract DAAH01-69-C-1812 for the Advanced Research Projects Agency. The report is organized into five sections. Section 2 lists the significant physical and manpower resources of the contract. Section 3 includes the contractual work statement and shows how each of the tasks has been met. Section 4 contains a discussion of the work plan under which the SDC team activities were organized. Section 5 covers the activities after the installation of the computer at MRDC.

2. RESOURCES

2.1 PHYSICAL RESOURCES

At the end of the contract period, the principal physical resources of the ARPA/MRDC Computer Laboratory consisted of the items listed below. The problems encountered in the production, implementation, and support of this equipment are discussed in detail in Section 4.4.

- An IBM 1800 computer with 32,000 words of high-speed memory and a two-microsecond cycle time, with the following attachments:
  - One Model 1442 card reader/punch.
  - One Model 1443 high-speed printer.
  - One Model 1627 plotter.
  - Two Model 2401 tape drives.
  - One Model 1810 disk storage device with two drives.
  - One Model 1816 printer-keyboard.
  - One Model 1896 start/stop communications adapter.
- The following consoles and devices for supporting console connections that interface with the Model 1896 communications adapter and are not supported by IBM software:
  - Four Model 2740 IBM consoles with limited distance data sets.
  - Two Shinko teleprinters. (These Japanese-made teleprinters print both Thai and English characters and are the standard teleprinter device used in the Thai civilian and military communication networks.)
  - One Model KSR33 teletype.
  - One acoustic coupler.
  - Three Lenkurt data sets.
  - Five SDC-produced signal converter devices.
  - Two Model 029 keypunches.

2.2 MANPOWER RESOURCES

Although minor changes were made throughout the life of the contract, a typical employee breakdown was as follows:

• U.S. Professionals	3
• Thai Professionals	
. Programmers and Programmer Trainees	
(Full-Time)	3
(Half-Time)	2
. Linguist (Half-Time)	1
• Thai Nonprofessionals	
. Secretary	1
. Computer Operators	2
. Keypunch Operator	1

3. PERFORMANCE OF CONTRACT TASKS

This section lists the tasks and subtasks of the contract work statement and indicates the manner in which each was successfully concluded.

3.1 TASK I

Task I. Increase the capabilities for handling the Thai language in a computerized environment; specifically:

- Interface two Thai-English teletypes with the computer in the "computer laboratory."

  This subtask was successfully completed.

- Design and procure an additional prototype Selectric typewriter Thai print element to provide maximum legibility.

  This subtask was eliminated after discussion with ARPA because the original prototype element had greater legibility than originally anticipated.

- Develop routines and programs required to input, sort, format, and output in Thai.

  Such routines have been produced and are discussed in more detail in Section 4.

- Develop routines to transcribe the Romanized equivalents of Thai words.

  Such routines have been developed and are discussed in Section 4, as well as being specified in detail in Appendix B.

- Promulgate necessary programming conventions so as to be useful as standards in the Thai computer user community and encourage the adoption of such standards.

  Our most significant effort in this area was the development of the Thai sorting algorithms. Since no programming standards group exists in Thailand, it was not possible to gain official acceptance of any of our internal standards, but a groundwork was laid for unofficial use by personal contact.

3.2 TASK II

Task II. Increase the technical capability of ARPA RDC-T/MRDC Computer Laboratory to:

- Develop a background of experience and knowledge of the problem inherent in the introduction of sophisticated computer technology into Thailand that is also applicable to technologically unsophisticated environments.

  This experience and knowledge were automatically gained from the implementation of the other subtasks.

- Train approximately three Thai nationals as systems designers and programmers competent in the efficient utilization of resource-sharing systems.

  Four Thai nationals received training as described. This subtask, while legally met, was not as successful as either SDC or the Government would have wished.

- Provide the necessary equipment and technical capability to fa familiarize senior officials of the Royal Thai Government with the capabilities and limitations of computer systems.

  SDC did provide such equpment and technical capability and, in cooperation with ARPA RDC-T, has achieved some results in which both organizations take pride.

- Provide necessary equipment and technical capability for performing developmental/experimental computer tasks.

  The production of a resource-sharing system and a Thai transliteration system was successfull.

- Provide technical capability and necessary equipment to train programmers and systems programmers other than the computer laboratory staff.

  The successful completion of this subtask is attested to by the fact that Chulalongkorn University did for some time train programmers utilizing SDC-provided equipment and SDC-trained instructors.

3.3 TASK III

Task III. Support the development of computer technology at Chulalongkorn University in the following manner:

- The resources of the computer laboratory will be made available to Chulalongkorn University representatives for 20 hours per month. The contractor will provide computer operators for this use but any special training required will be provided by Chulalongkorn University.

  The time was provided.

- Provide the computer resources and necessary training in their use to Chulalongkorn University representatives. The intent of this subtask is to channel the benefits derived from Task II into Task III as rapidly as possible, but to do this without unduly affecting the activities of Task II.

  SDC provided such resources and training as desired by Chulalongkorn University.

4. ORGANIZATION OF WORK

Most of the activities of the contract can be categorized as one of the following:

- Operations of the computer and its associated "service bureau."
- Production of "utility" programs.
- Production of a resource-sharing system.
- Production of a system for Thai transliteration.
- Production and maintenance of hardware and software for attaching the various consoles.

Production is intended in this instance to include necessary planning activities. Training is not specifically included, since on-the-job training was provided in all the activities mentioned. Administrative problems naturally made significant time demands on the SDC staff, but these will not be discussed.

4.1 COMPUTER AND "SERVICE BUREAU" OPERATION

Problems of maintaining the computer, training operators, making documents available to users, and so forth were solved by reasonably standard methods. Specific hardware maintenance problems are discussed in Section 4.4.

4.1.1 Production of Utility Programs

On receipt of the IBM 1800 computer it was discovered that the IBM "Card System" programs were totally inadequate for any sort of program-development effort. Use of the more complex disk-based systems was ruled out on the basis of the extra time required to get the systems operating in Thailand, combined with the lack of adaptability of available programs for use in the projected SDC resource-sharing system.

Thus, during the first several months of the contract, the SDC staff produced a series of new programs that were necessary for future work. In most cases the programs were created from scratch; in a few cases, most notably the assembler, the IBM programs were modified. A representative list of such programs includes:

- Card to Tape.
- Tape to Printer.
- Tape Duplicate.
- Disk to Printer.
- Assembler Utilizing Tape Buffer.
- Read In System Using Tape Library.
- Simple Console Controlled Operating System.

The programs were implemented as quickly as possible and had few refinements and little documentation. They did serve as excellent training programs for some of our trainees.

4.2 PRODUCTION OF A RESOURCE-SHARING SYSTEM

This system was one of the most complex of its type ever worked on in Thailand. The system, however, must be considered incomplete since certain highly desirable capabilities, such as sophisticated debugging techniques, could not be implemented in the allowed time. However, even in this incomplete state, demonstrations of its present capabilities have had a significant impact on the opinions and planning activities of those who have seen them. In addition to the production of the resource-sharing system itself, the staff also made massive modifications to the IBM-produced FORTRAN IV program so that it would work in an interactive mode under the resource-sharing system and could utilize Thai characters on both input and output. (The Thai language capability is usable only in connection with the Shinko teleprinters.)

In its present form the system supports the following functions:

- Six consoles can be used independently at the same time.
- Any program, whether originally written in FORTRAN or Assembly language, that meets size restrictions and does not conflict with I/O restrictions can be run at any of the consoles.
- All I/O devices can be in simultaneous operation without disrupting any resource-sharing function.

The FORTRAN IV program, which is one of the programs available for operation at any console, has all of the capabilities shown in the IBM specifications plus the following additional attributes:

- A comprehensive editing capability allows programs to be written, listed, and modified on-line.
- Thai characters pass unchanged through the various conversion subroutines.
- Editing, compilation, and program operation may all be requested in an interactive mode.
- All standard subroutines are available and are added to the program during the "read in" process.
- Complete compatibility exists with the IBM 1800 FORTRAN in the sense that all features described in the IBM documentation still exist unmodified. All changes involve additional capabilities or changes in methods of operation not of concern to users.

In addition to the effort involved in writing the programs, the staff spent considerable time in teaching the correct operation of the consoles, the system, and the compiler. Rough documentation was provided, but the Thai environment requires more personal attention to users' problems than that necessary in environments where such systems are now almost taken for granted. Therefore, most instruction involved direct teacher-student interaction.

As previously indicated, the focus of concern in this project was in demonstrating the possibility of producing a system in the Thai environment and using it to demonstrate the potential of interactive computing to Thais. No claim is made that the project itself represents any significant new departure in the design of such systems. The design is comparable to previous systems and, hence, is not discussed in detail here. For those interested in a more technical exposition, Appendix A contains a list of subprograms and other pertinent data.

4.2.1 Description of TSS

During a TSS run the system itself resides in core from address /0000 to /3F80, or approximately the first half of core. At any given time, the currently executing object program will reside in core from /4200 to the end of the program, which must be equal to or less than /7FFF, the end of core. The intermediate area /3F80 to /41FF is used for transfer vectors that connect an object program to its subroutines and for the object-program/system interface routine associated with this object program. The area /3F80 to the end of the object program participates in swaps on a round-robin basis.

The disk pack on disk drive #0 is commonly called the system disk. It contains a catalog of its contents by name and location, all available object programs in core-image format, the TS system in core-image format, and the subroutine library in IBM compressed-deck format for use by DLØDE. The beginning of the subroutine library file is a catalog of subroutine names, alternate entry points, and locations on the disk.

The disk pack on disk drive #1 is the object program disk or swap disk. It contains an area for each terminal in the system. Each area is divided into two sub-areas, one for the core image of the object program, and one for the use of the object program. In the case of the FORTRAN compiler as an object program, the latter sub-area is further divided into a source-program file, a list file, a symbol-table file, and a binary-output file. During a TSS run the object program will be moved from its disk file to core each time its turn comes up and it is ready to execute, and then moved back from core to its disk file when its time is up or it makes a system request (i.e., Requesting Terminal I/O) and some other terminal is ready to execute.

4.2.2 Preparation of the System

Three programs--DPREP, SYSLD, and NITAL--were written for off-line preparation of TSS. DPREP must be executed whenever it is desired to make a new system disk for disk drive #0 or change the current disk. It loads a TSS object program into core and then moves the core image of that program to disk drive #0. DPREP will accept input from tape or card reader and will make a new disk or add to an existing disk. If an input program has the same name as one currently on the disk, the current one will be replaced by the new one. In any case, DPREP will finally put a new catalog on the disk.

SYSLD must be executed whenever changes larger than patches are made to the TSS. SYSLD creates a core image of the TSS by taking commands from the 1816 typewriter regarding identity of next input. Inputs may be mainline programs, subroutines, patch decks, and libraries of subroutines. When it is finished, the core image is put on disk drive #0 and an entry is added to the catalog.

NITAL is the initialization program for TSS. It is loaded and executed each time a TSS run is desired. NITAL loads TSS into core, prepares the TSS DATA area, reads and adds any patches to the system, prepares the swap disk by adding the object-program/system interface routine with terminal-specific information to each terminal's disk area, reads the object-program catalog and subroutine catalog into core, and transfers control to TSS.

4.2.3 System Components

The following major programs constitute the system:

- SCHED - Schedules the operation of all other components. The system flow diagram is basically of SCHED.
- TELIØ - Initiates terminal I/O by calling SHIN for Shinkos and C2740 for 2740s, and does the checking for completion of terminal I/O.
- SWAPR - Moves operating programs from disk to core and vice versa, and saves and sets up environments (e.g., accumulator, index registers).
- LODER - Searches the catalog of object programs for a requested name and, if it finds the named program, transfers its core image from disk drive #0 to the appropriate core-image area on disk drive #1 for the requesting terminal.
- DLODE - Loads the binary output of an on-line FORTRAN compilation from the requesting terminal's disk drive #1 binary area into core, adds any subroutines called from the subroutine library on disk drive #0, and prepares that program to execute.
- COMND - Processes commands from the terminal to the system. /LOAD AAAAA, /LINK NN, /UNLINK NN, and /RESTART are implemented.
- DATA - Large common data area available to all system programs.

4.2.4 Object Program/System Interface

CNTRL is a small interface routine added to each terminal's core-image area on disk drive #1 by NITAL, the TSS initialization program. It contains terminal-specific information available to both the object program and the system. CNTRL processes all requests made by the object program to the system such as terminal I/O, load another program, stop, or pause.

4.2.5 Object-Program Preparation

Since TSS has no debugging capability for an object program, checkout of such a program must be done outside of TSS. Once a program has been checked out it can easily be added to the system disk. It must either be originated at /4200 or assembled or compiled in relative mode. The resultant compressed deck from the assemble or compile is converted by a modified version of the IBM core-image converter, which will place it at /4200. In addition, the deck of library subroutines that have been modified specifically for TSS must be included in the conversion. In every case these subroutines have identical names to those that were used for checkout in the stand-alone mode. These subroutines and their functions in TSS are as follows:

- PAUSE - This version of PAUSE fields FORTRAN I/O errors and prints appropriate error messages on the terminal.
- DISKN - Performs disk I/O in TSS by considering calls to be relative to the calling terminal's disk area.
- PRNT1 - Corrected to handle 1443 output in TSS for assembly-language programs.
- PRNTZ - Corrected to handle 1443 output in TSS for FORTRAN programs.
- SHINZ - The routine called by FORTRAN programs for terminal I/O. It is the parallel of PRNTZ, TYPEZ, etc., in that it processes I/O based on FORTRAN format statements. It differs in that input need not be column conscious.
- TRMIØ - Called by SHINZ to interface with CNTRL.

In addition there are several dummy routines that merely replace their name namesakes, save parameters, and branch to CNTRL for interface with the system. They are: SHIN, C2740, KSR33, SPIRN, GETPG, and STØP.

4.3 PRODUCTION OF A SYSTEM OF THAI TRANSLITERATION

A relatively nontechnical description of the transliteration system is given in this section. For a more detailed discussion of the rules written in TRANS, the character-oriented, string-processing language designed for use by the transliteration system, see Appendix B. For a technical discussion of TRACTS (Thai/Roman Computerized Transliteration System), see Appendix C.

This project wrote a set of computer programs to transliterate words written in Thai to a Romanized version of these words that can be read by those who are not familiar with the Thai language or alphabet. Transliteration standards have already been established by the Thai Royal Institute and have been used in applications requiring accurate and consistent rendering of Thai geographic names. Unfortunately, these standards are of use only to someone who is already familiar with the pronunciation of the Thai word to be transliterated. This, then, was essentially a research project to determine the extent to which transliteration can be performed mechanically (by a computer) given only a Thai word as input.

4.3.1 General System Description

The main component of this system is an IBM 1800 computer with a 32K, 16-bit word memory, using a Shinko teleprinter as an on-line input/output device. This teleprinter is capable of printing both Thai and Roman characters. In addition, an IBM Model 2401 tape drive and 1810 disk are used for auxiliary I/O.

At the outset the work was divided into two efforts: Thai orthographic research and computer programming. As an articulation between the two, a string-processing language, TRANS, in which the orthographic/transliteration rules, when discovered, could be expressed and tested, was devised. This allowed work to proceed on both fronts simultaneously. While research was conducted to determine what rules eventually would be written, an interpreter program was being developed to translate and execute the TRANS rule statements. The string-processing language and the on-line characteristics of the system permitted rule changes and additions at every stage of the testing process without reprogramming.

4.3.2 Linguistic Research

The ultimate goal of the linguistic research was to determine the pronunciation of a Thai word based on its spelling. This work fell into six main areas:

- Preposed vowel placement. Five vowels in the language are written in front of the consonant after which they are pronounced. In some cases the vowel is pronounced after the first consonant; in other cases it is pronounced after the second of two following consonants. A related problem concerns the juxtaposition of several separated characters that are used to represent a single simple vowel or diphthong.

- Determination of syllable boundaries. The majority of the rules in the system are concerned with discovering syllable boundaries. A great variety of clues are employed for this.

- Insertion of implicit vowels. Many words have syllables pronounced with a short A or O vowel, for which there is no explicit representation in the written word.

- Reduplication. Certain syllables or portions of syllables are pronounced twice but written just once.

- Miscellaneous. Most of the other phenomena are relatively easy to handle once the first four, particularly syllable boundaries, have been determined. These other phenomena concern, for example, silent characters (characters that are written but not pronounced) and characters that are variously pronounced depending upon their position in the syllable.

- Exceptions. Words or groups of syllables that do not conform to the rules devised for the above groupings of words.

4.3.3 The Program System

Most of the system's functions are initiated by commands typed on-line. A control program checks the legality of the command, calls and operates the appropriate programs, and initiates a read operation to accept a new command. These commands affect the translation of rules, the input of data to be transliterated, the execution of a single rule or sequence of translated rules, the maintenance of the rule file, the alternate acceptance of commands from the card reader, and the mode of system operation (debugging or demonstration mode). In the debugging mode, intermediate output is provided to help determine the cause of any errors. In the demonstration mode only one command is accepted, that defining the Thai word to be transliterated, and only one output is given, the Romanization.

The most important function concerns the TRANS rules. There are three essential parts to a rule: a rule name, a string description, and a string change or changes. There is only one data area in the language. All rules operate on this one string. Any changes made are made to this string. When a rule is executed, the string description is compared with the string in the data area. If there is a successful match, the string changes are performed. Provision for selective string changes (according to the manner in which a match was performed), conditional GOTO statements, and exceptions are provided in a fourth, and optional, portion of the rule. Provisions for extensions to the language have been made so that special conditions on rule symbols in the string description can be naturally and easily invoked. Rules expressed in the TRANS language are input via Shinko teleprinter or cards. They are translated to table entries and normally saved in core or on tape. Upon subsequent command a rule is executed; that is, the entries are input as parameters to table-driven subroutines that perform the match with, and changes upon, the data string.

Certain words and syllables in the Thai language can be transliterated correctly by our system only if they are treated as exceptions to the TRANS rules. Accordingly, we have a separate program that reads these exceptions from cards and files them on disk. This disk is subsequently used by the operational transliterating system. While the TRANS rules are operating on the Thai word that was input, the disk is being read to determine if the word contains any exceptional syllables. Any exceptions found are replaced by a new string of characters that will enable the TRANS rules to correctly transliterate the word. If the entire input word is exceptional, the TRANS rules can be avoided entirely and the input can be replaced by a string that is capable of being transliterated directly.

An additional component, an interpretive TRACE program, was written to check IBM 1800 machine-language programs to determine illegal address references. This TRACE program was used in the early stages of the programming effort.

4.3.4 Final Status

As of 30 April 1971 the system contained 175 TRANS rules and 1,800 exception sequences. Tests using words selected at random from the dictionary showed results ranging from 90% to 96% correct, with the higher percentages occurring at the end of the contract period, as final rules and exception modifications were made.

4.4 ATTACHMENT AND MAINTENANCE OF CONSOLES

Three different types of consoles are now supported by the resource-sharing system, and a large number of methods of connecting remote consoles to the computer are provided. In tabular form, the possibilities are as follows:

Console Type	Transmission Line
IBM 2740	Private line, 4 conductor, limited distance.
Shinko Teleprinter	±100 volt DC line. Public telephone lines. Private lines. Thai Telex network.
KSR33 Teletype	Public telephone line. Private line.

4.4.1 IBM 2740

The IBM 2740 is connected through IBM equipment exclusively. It uses a 7-bit code unlike any other console and is not supported by any software for use on the IBM 1800 computer. These were the first consoles connected. The principal problem involved in supporting them was a poor understanding of the working of the 1896 Model communications adapter, which is made by Astrodata Corporation and is not well described in the available literature. The consoles are always in a slave mode (i.e., they can only reply to a communication, they can never initiate a message), so only one console need be handled at a time by the system. This makes possible a relatively small program that is concerned primarily with station control and character transformations.

4.4.2 Shinko Teleprinter

The Shinko teleprinter is an unmodified device intended for use in a teleprinter network. It has a three-shift keyboard that provides all Thai characters, numbers, English capital letters, and some special symbols. It has a 6-bit code and operates at 50 baud. It is totally incompatible with any American console. SDC designed and built signal converting devices to convert the ±100-volt DC to EIA standard computer interface voltages and connectors. The IBM (Astrodata) communications adapter provided adjustment to the 6-bit code and the 50-baud rate. Programs were written, using interrupt circuits, that provided for simultaneous operation of two Shinko teleprinters. All code was converted to a special EBCDIC containing Thai on input, and from EBCDIC to the proper Shinko code on output. Proper case shifts are inserted as appropriate on output. An additional device was designed and constructed by SDC to permit connecting the computer to the local teleprinter network. However, the requested connection to the system was never made because of delays in installation of the line. (See Appendix D for a technical description of the SDC-supplied hardware.)

4.4.3 KSR33 Teletype

A portable version of the KSR33 teletype was obtained to provide console operation from distant, temporary stations. The KSR33 character set and baud rate are different from any of the other consoles. The Model 1896 was adjusted to provide for the KSR33 and a program was written to integrate it into the system.

4.4.4 Maintenance Problem

Maintenance of the consoles and the computer circuits concerned with the consoles was a considerable problem. There are no IBM maintenance programs that can be used to diagnose problems in the communications adapter as it is set for our consoles. SDC wrote programs to perform this function and assisted IBM maintenance engineers when questions of malfunctions occurred with respect to this equipment. The maintenance of non-IBM equipment is a greater problem. The Shinko teleprinters are maintained by the local importer. The SDC-produced signal converters are difficult to maintain with available personnel. However, spares are available, and non-functioning devices can be returned to the United States for repair. The greatest maintenance problem is the KSR33. No trained repairman or spare parts exist in Thailand. Even minor maladjustments caused severe problems.

5. RECENT ACTIVITIES

The agreement with Chulalongkorn University ended October 30, 1970. The computer was then shut down and moved to space provided at the Military Research and Development Center. All office equipment and personnel were also moved. The computer began operation in the new location in December. Between that time and the computer's removal in May 1971, activities involving SDC personnel and the IBM 1800 took the form of demonstrations, communication tests, computer usage by ARPA-authorized users, classes, and seminars.

SDC and ARPA coordinated in presenting demonstrations for numerous groups including representatives of the Thai government and military, US military, ARPA, MRDC, and local educators. These demonstrations exhibited both the hardware and software systems and included examples of transliteration, the resource-sharing system and a cryptography program written by a member of MRDC.

Communication tests were performed using the IBM 1800, the KSR33, the resource-sharing system, and both military and civilian telephone lines. Successful tests were made from many locations in Bangkok.

With the authorization of ARPA, many individuals made use of the facilities of the computer center. These users included representatives from MRDC, the Thai military, ARPA contractor personnel, as well as individuals from the classes and seminars.

SDC personnel conducted classes for Thais recommended by ARPA. These classes covered a wide range of topics in contemporary computer state-of-the-art. One class consisted of surveys of general programming, multi-programming, multiprocessing, timesharing, machine languages, higher-order languages, and debugging, and gave the students hands-on experience in many of these subjects with the hardware and software available at MRDC.

Another class concentrated on general programming and IBM 1800 assembly language. Again, the computer facility was used for hands-on experience. (See Appendix E for Class Outlines.)

In addition to these formal classes, seminars were offered and individuals were tutored by SDC personnel. Seminars were conducted for three groups of Thai military personnel. These groups were encouraged to discuss their plans for computer usage and were offered guidance and suggestions on technical matters.

In May, SDC participated as needed with IBM, ARPA, and MRDC to remove and ship the IBM 1800 computer. From the time of the computer's removal from Bangkok in May until the contract ended on 30 June, SDC continued its role of consultant, teacher, and adviser, and continued the documentation effort for the transliteration program.

APPENDIX A

A RESOURCE-SHARING SYSTEM DESIGNED TO OPERATE IN THE THAI ENVIRONMENT

1. TSS

1.1 SYSTEM PROGRAMS

SCHED	schedules the operation of all system components.
TELIO	initiates terminal I/O by calling CHIN for shinkos and C2740 for 2740's and does the checking for completion of terminal I/A.
SWAPR	swaps operating programs in and out of core, saves and sets up environments.
LODER	transfers programs from the library of available operating programs to the core image area for a given terminal.
DLODE	loads the binary output of a FORTRAN compilation into core, adds the required subroutines and prepares it to execute.
CØMND	processes commands from the terminal to the system. LOAD XXX, LINK NN, UNLIKN NN and RESTART are implemented.
DATA	large common data area available to all system programs.
CNTRL	small interface routine added to all operating programs by the LODER. Contains terminal specific information.

1.2 SYSTEM PREPARATION PROGRAMS

DREP	loads the library of available operating programs onto disk in core image format or adds programs to disk and builds a catalog of those programs.

SYSLD	creates a core image of the TSS by taking commands from the 1816 typewriter regarding identity of next input. Inputs may be mainline programs, subroutines, patch decks, and library of subroutines. When finished it saves the core image on the same disk as the library of available operating programs and adds an entry to the catalog.
NITAL	prepares the system disk used for swapping, loads the system in core and transfers to it. In addition NITAL prepares the system DATA area.

1.3 AVAILABLE OPERATING PROGRAMS

FORTRAN compiler (6 passes)

EDF	FORTRAN editor. Accepts input from a terminal and prepares a disk file. Using EDF one may write a FORTRAN program on line, modify it, list it, add to it, delete from it, etc. The disk file created is input for the FORTRAN compiler.

Several demonstration programs of a non-technical nature.

DEBUG AIDS

DK*PR - lists an EBCDIC disk file.

DK*CD - punches from a disk file.

DDUMP - dumps in hexadecimal a disk file.

2. FORTRAN

2.1 ADDED CAPABILITIES

- Full EBCDIC - allows any character in FORTRAN FORMAT statements, i.e., Thai characters.
- Symbol table output - will be used later by the TSS debugger.
- Addition of terminal I/O as a FORTRAN logical unit.
- Floating format for terminal I/O so user need not be column conscious.
- FORTRAN editor (EDF) (described in TSS).

2.2 FORTRAN COMPILER DESIGN CHANGES

- Passes - reduced to 6 from original 26.
- Disk I/O added.
- Made interactive via terminal I/O.
- Load of passes from tape automated for batch system.

	FORTRAN BINARY	INPUT	GENERATED BINARY	LISTING
ORIGINAL (26 PASS)	CARD	CARD	CARD	PRINTER
BATCH (6 PASS)	TAPE	CARD	CARD	PRINTER
INTERACTIVE (6 PASS)	TAPE	DISK	DISK*	DISK & PRINTER
TSS (6 PASS)	DISK	DISK**	DISK*	DISK

*Available for Loading or Punching via EDG.
**Input via EDF from card or terminal to disk.
Available for listing on printer or terminal via EDF.

APPENDIX B

TRANS: A SPECIAL PURPOSE CHARACTER-ORIENTED STRING PROCESSING LANGUAGE

PREFACE

This appendix presents all of the information needed to understand rules or systems of rules written in the TRANS string processing language. TRANS was designed to be used primarily in TRACTS, a computerized system for transliterating words written in Thai script to a Romanized version of these words. This totally non-numeric language is composed of rules that manipulate individual characters of a single data string according to detailed matching specifications.

TABLE OF CONTENTS

GENERAL INFORMATION ABOUT RULE OPERATION

TRANS is a special-purpose character-oriented string processing language.* The basic unit of computation in the TRANS language is a rule. There are three essential parts to a rule, a rule name, a string description, and a string change or changes. There is only one string-data area in the language. All rules operate on this one string. Any changes made are made to this string.

The canonical form of a rule is as follows:

```
RULE  rule name R     . string description = string changes.  exceptions and goto.
            optional                                                  optional
```

When a rule is operated, the string description (abbreviated SD) is compared with the string in the data area. If the SD successfully matches the data string, each string change formula of the rule is operated once. If the SD does not match the data string, the string changes are not operated.

*The "special purpose" for which TRANS was designed is to transliterate words written in the Thai language to a Romanized version of these words. With a few minor changes the language could have more general applicability. TRANS is character-oriented in that all changes made to a string must be specified in terms of a single character. For ease of understanding by English speaking readers, all examples in this paper are written using Roman characters exclusively.

The SD is specified in terms of variables and individual characters of the string.

```
(1)          .  X  *1H X  =  *1=0  .

     string description  string change

             SD                 SC

(2)  P R A H M A  data string before the rule in (1) is operated

(3)  P R A M A    data string after the rule in (1) is operated
```

The SD in (1) above specifies that a successful match for this rule would be a string in the data area that may contain any number of characters but must contain at least one character H. The single string change formula (1) specifies that the character labelled *1 by the successful match of the SD and the data string is to be erased. In this case the character labelled *1 is H. Thus H is erased.

The comparison between the data string and the SD of the rule is made from left to right one character at a time. In reference to the rule in (1), if there is more than one H in the data string, the leftmost one only will be erased.

Some rules are repeatable. This means that after one cycle of the rule (successful match followed by one operation of each string change formula) other cycles are attempted. A repeatable rule continues operation until the string description fails to match the data string.

(4) R . X *1H X = *1=0.

The character R in front of the string description portion of a rule makes the rule repeatable. The rule in (4) above is identical to the rule in (1) in every respect except repeatability. Note below the difference in the operation of rules (1) and (4) on data string (5).

(5) H U H N I G H G A N example data string

(6) U N I G G A N data string (5) after it has been changed by the repeatable rule in (4).

(7) U H N I G H G A N data string (5) after it has been changed by the nonrepeatable rule in (1).

DETAILS OF THE STRING DESCRIPTION

The string description must account for each and every character in the data string in order to be considered as having successfully matched the data string. Normally it is not possible or desirable to account for each character by specifying it exactly. A string description such as (8) would successfully match only one data string, that string consisting only of the three characters J, A, and N in that order.

(8) . J A N =

Variables

To describe the data string fully yet allow the SD to apply to a wide range of strings, TRANS provides two types of variables, X variables and Z variables.

The X variable, which is represented in the SD by the single character X, may represent or account for any number of consecutive characters in the data string, including no characters. A string description consisting of only the variable X would, in fact, successfully match every data string. The X variables in the SD part of the rule in (1) permit the rule to successfully match any data string as long as it contains the character H, regardless of the kind and number of characters that precede H or the number and kind of characters that follow it. When the rule in (1) is applied to the data string in (2) the first X accounts for the characters P R A. The second X in (1) accounts for the characters M A.

The SD in (9) below will successfully match any data string that <u>begins</u> with the character H.

```
(9)      . H X =
```

Since X may also represent zero characters in the data string, the SD (9) successfully matches a string that consists of only the single character H.

The SD (10) will successfully match any data string whose last (right most) character is H. This includes the one character string, H.

```
(10)        . X H =
```

Only five of the seven string descriptions in (11) below will successfully match the data string in (2), P R A H M A.

```
(11a)        . X A       =
(11b)        . P X       =
(11c)        . P R X     =
```

*(11d)	. X R X H	=
*(11e)	. X R X H A	=
(11f)	. X R X H X	=
(11g)	. X R A X M A	=

String description (11 d) does not successfully match data string (2) because it fails to account for the final characters M A. String description (11e) fails to account for M. It specifies that the character A be the final character of the string and that it immediately follow H.

The Z variable has considerably less scope than the X variable. The Z variable matches any one,but only one,character in the data string. Whereas X may account for zero characters, Z must account for one character. Consider the following examples:

(12) T A R A G example data string for SDs (13a)-(13j).

	string descriptions		1st(left most) Z represents	2nd Z represents
(13a)	. Z X	=	T	-
*(13b)	. Z Z	=	string description does not match (12)	
(13c)	. Z Z X	=	T	A
(13d)	. X Z	=	G	-
(13e)	. X Z Z	=	A	G
(13f)	. T A Z X	=	R	-
*(13g)	. X G Z	=	string description does not match (12)	
(13h)	. X Z Z X	=	T	A
(13i)	. X A Z X	=	R	-
*(13j)	. X R Z	=	string description does not match (12)	

String description (13b) will only match a data string that consists of exactly two characters. In attempting to match SD (13g) we proceed from left to right in the data string looking for the character G. When it is found, the X preceding G represents the characters T, A, R, A. But (13g) also specifies that one character must follow G. Since no character follows (13g) does not successfully match the string in (12). SD (13j) also fails to match data string (12). In this case we are looking for any one (but only one) character after R. Since there are two characters after R in the data string, (13j) fails to match (12).

Further explanation must be given here about the method of accounting for characters in the data string by means of an X. Matches are attempted by assigning as few characters as possible to each X as it is encountered from left to right. Thus, in SD (13h) it is conceivable that the two Z's represent or account for any two consecutive characters in (12) as follows:

	X	Z Z X
(14a)	-	T A R A G
(14b)	T	A R A G
(14c)	T A	R A G
(14d)	T A R A G -	

However, by the rule of assigning as few characters as possible to each X as it is encountered from left to right, TRANS accepts the first correct allocation encountered, (14a). Similarly, SD (13i) could account for each character in (12) in two ways:

```
          X   A Z X
(15a)     T   A R A G
(15b)     T A R A G -
```

Following the rule stated above, the allocation of string symbols to characters in the data string is as shown in (15a).

Choice Brackets

Choice brackets are used at a place in the SD where it is necessary to express the acceptability of any one character or string of characters from a set that includes all of the possible choices at that point. The choices are enclosed between parentheses and are separated from each other by commas. The SD (16) below with an initial choice bracket successfully matches data strings (17a) and (17b).

```
(16)      . ( F, T ) A J =
(17a)       F A J
(17b)       T A J
```

String description (18) successfully matches strings (19a), (19b) and (19c). Data string (19d) is not successfully matched. Within a choice bracket one choice must be made.

```
(18)      . A ( B, C, D ) F G  =
(19a)      A B F  G
(19b)      A C F  G
(19c)      A D F  G
*(19d)     A   F  G
```

String description (20) with two choice brackets successfully matches strings (21a), (21b), (21c) and (21d).

```
(20)     .  A ( B, C ) ( D, E ) F G  =
(21a)      A   B  D   F G
(21b)      A   B  E   F G
(21c)      A   C  D   F G
(21d)      A   C  E   F G
```

From among the choices within each choice group only one may be chosen. Thus (21e), (21f) and (21g) are not successfully matched by string description (20).

```
*(21e)     A B C D F G
*(21f)     A B C D E F G
*(21g)     A B D E F G
```

All of the previous examples concerning choice brackets have presented choices among single characters. However, the language allows one to specify choice from among strings of characters also. Consider SD (22) below.

```
(22)      . A ( B, C D, E F G ) H  =
```

String description (22) will match the three strings below.

```
(23a)     A B H
```

(23b) A C D H

(23c) A E F G H

Choice brackets may be nested within other choice brackets to any depth. Strings (25a) through (25d) below are the strings that are successfully described by SD (24).

(24) . A (B, C (D, E), F G) H =

(25a) A B H

(25b) A C D H

(25c) A C E H

(25d) A F G H

String description (26) successfully matches only those strings shown in (27a) - (27f).

(26) . A ((B, C) (D, E), F, G) H =

(27a) A B D H

(27b) A B E H

(27c) A C D H

(27d) A C E H

(27e) A F H

(27f) A G H

String description (28) is almost the same as SD (26) except that it contains an extra set of parentheses. However, these parentheses serve no function whatever and in addition will cause errors in the matching procedure.

(28) . A (((B, C) (D, E)), F, G) H =

In order to be meaningful a set of left and right parentheses must contain at least one comma that is not enclosed within a deeper set of parentheses. It is for this reason that SD (28) above and (29a) below are incorrect.

```
*(29a)    . A ( ( B C ( D, E ) ), F, G ) H  =
```

SD (29b) is matched by strings (30a) - (30d).

```
(29b)    . A ( B C ( D, E ), F, G ) H  =

(30a)      A B C D H

(30b)      A B C E H

(30c)      A F H

(30d)      A G H
```

String description (31) containing the most deeply nested choice bracket we have discussed thus far is matched by the following seven strings, (32a) - (32g).

```
(31)     . A ( ( B, C ) ( D, E ( F, G ) ), H )  =

(32a)      A B D

(32b)      A B E F

(32c)      A B E G

(32d)      A C D

(32e)      A C E F

(32f)      A C E G

(32g)      A H
```

Optionality

Let us suppose that we want to perform a string change on any string in which the character A immediately precedes the character C. The SD would, of course, be represented as in (33).

```
(33)      . X A C X  =  string changes.
```

Let us further suppose that we want to perform the same string changes as (33) if the character B occurs between A and C. We could handle this by writing another rule (34), which when combined with (33) would operate on all strings that contained the sequence A C or A B C.

```
(34)      . X A B C X  =  string changes.
```

We say then for this phenomenon that character B is optional. We want to perform a particular set of string changes on a string that contains A C, where the character B is optionally present between them.

Optionality is described in TRANS by the character % as the last choice within an option bracket. Thus, SD (35) is the equivalent of the combination of string descriptions (33) and (34).

```
(35)      . X A ( B, % ) C X  =  string changes.
```

Another example of optionality is presented in SD (36), which is the same as SD (31) except for the addition of the optionality symbol after the character C. With the addition of the optionality symbol, 3 more strings (37a) - (37c) will be matched. So SD (36) is matched by strings (32a) - (32g) and (37a) - (37c).

```
(36)      . A ( ( B, C, % ) ( D, E ( F, G ) ), H )  =
(37a)       A D
(37b)       A E F
(37c)       A E G
```

STRING CHANGES

If the SD part of a rule correctly matches the data string, the string change (SC) part of the rule is operated. The SC is divided into formulas. There may be any number of formulas in the SC, but there must be at least one. SC formulas are separated from each other by the colon character (:). The last SC formula is terminated by a period. Each SC formula will alter just one character or character position of the data string. A SC formula cna insert a new character into the data string, change a character in the data string or erase a character of the data string.

When a correct match between the SD and data string is obtained, each constant and Z variable that participated in the correct match accounts for and points to a single character in the data string. The data string character that is pointed to is denoted internally by its position in the data string, where the leftmost character is indicated by a 1.

```
(38)     A B C A E A  data string

         1 2 3 4 5 6  position number

(39)     X B Z X A = string description

           2 3   6    position number this constant or Z variable
                      points to after successful match of SD (39)
                      and data string (38)
```

Any constant of Z variable in the SD of a rule may be labelled. A label consists of two characters, an asterisk (*) and a single numeral digit, 1 through 9. The label immediately precedes the constant or Z variable it denotes.

String change formulas are specified in terms of labels; for example:

(40) *1 = 0.

String change (40) says to delete that character in the data string that is pointed to by the SD symbol that is labelled *1. Thus, if the data string is as represented in (38) and rule (41) is executed, the 6th character of the data string is deleted from the string, leaving A B C A E.

```
(41)     . X B Z X *1A  =  *1=0.
             2 3     6   position number in data string (38) that is
                         pointed to after successful match with the
                         rule in (41).
```

Again assume (38) is the data string. The rule executed is (42).

(42) . X B *1Z X *2A = *1=0: *2=0.

After the execution of (42), two characters are deleted from (38) and it appears as A B A E.

Neither the X variable nor the optionality symbol (%) may be labelled.

The same label may be attached to more than one character within a choice bracket, as in (43a) and (43b).

(43a) . *1 (A, B C, D) X = *1=0.

(43b) . (*1A, B C, D) X = *1=0.

A label immediately in front of the choice bracket or in front of the first character of a choice bracket denotes each single character choice or the left-most character of a string choice. Thus the label *1 denotes A, B, and D in (43a) and (43b). Since BC is not a single character choice, the label is only attached to B, the leftmost member of the string choice.

In SD (44) the label *1 is attached to A, B, D and E. Note that both A and B are leftmost characters of a string choice, while C and F are not.

```
(44)     *1 ( ( A, B ) C, D, E F ) X  =  *1=0.
```

New characters may be inserted into the data string either to the left or right of the character pointed to by a labelled rule symbol.

```
(45a)    *1 = M + *1.
(45b)    *1 = *1 + M.
```

SC (45a) will insert a new character, M. immediately to the left of the character pointed to by *1. SC (45b) inserts an M immediately to the right of the data string character pointed to by *1.

The rule in (47) will change the data string in (46) to appear as in (48).

```
(46)     A B A D E
(47)   . *1 ( R, Z B, T ) *2 Z X  =  *1=0:*2=F + *2.
(48)     B F A D E
```

The struccure changes allow a character of the data string to be duplicated and inserted elsewhere in the string. If the data string is as in (46) and the rule is as in (49) the resulting data string is shown in (50). The rule in (49) says that if you find the character B in the data string and there are at least two characters following it, make a duplicate of the second character following B and insert it in the data string to the left of (in front of) B.

```
(49)     . X *1B Z *2Z X  =  *1=*2 + *1.
(50)       A D B A D E
```

the 4th character of the data string by a duplicate of the second character.

It is possible in the language for one rule to perform different sets of string changes depending upon how the SD matched the data string. This is made possible by the assignment of different labels to the members of a choice bracket.

From the discussion of labels on page B-14, a label appearing in front of a choice bracket is assigned to each leftmost member of each choice. However, there is an exception to this; the label is not assigned to any symbol that is already labelled (i.e., that is immediately preceded by another label). Thus, in (55) the label, *1, is assigned to A and D but not to B, which is already labelled *2.

```
(55)      . *1 ( A, *2 B C, D E ) F X  =  *1=0:*2=K.
(56a)            B C F Y L
(56b)            K C F Y L
```

Now, when (55) is operated and (56a) is the data string, a correct match is found such that the second choice within the bracket, i.e. *2 B C, is used. Subsequently, when the string changes are operated, the first formula, *1=0, has no effect, since neither the A nor the D (the only symbols labelled *1) participated in the successful match. Thus, *1 does not point to any character in the data string and so any SC formula containing *1 has no effect. After (55) has operated on data string (56a) the data string appears as in (56b).

However, when (55) operates on data string (57a) the SC formula referring to *2 has no effect. The result is string (57b).

(57a)	D E F Y L
(57b)	E F Y L

In order for a particular SC formula to be meaningful, <u>all</u> of the labels in that formula must point to a character in the data string. This is further illustrated in rule (58) which says: starting from the left of the data string, look for a character C. If it is followed by at least two characters, the first of which is A or B, then duplicate these two characters and attach them immediately to the <u>right</u> of C in the order in which they appear. But if there is only one character following C, or if there is more than one but the first is not A or B, attach this character immediately to the <u>left</u> of C.

```
(58) .X *1C ( *2 ( A, B ) *3Z, *4Z ) X = *1=*1+*3: *1=*1+*2: *1=*4+*1.
                                         formula 1  formula 2  formula 3
```

The various possibilities are realized when the rule in (58) is operated on data strings (59a), (60a) and (61a).

(59a)	M N C B T L	initial string
(59b)	M N C T B T L	after formula 1
(59c)	M N C B T B T L	after formula 2 (formula 3 does not operate)
(60a)	Q C V W O	initial string
(60b)	Q V C V W O	after formula 3 formulas 1 & 2 do not operate
(61a)	C A	initial string
(61b)	A C A	after formula 3 formulas 1 & 2 do not operate

Abbreviation in the String Description

In the application for which we have used TRANS we frequently find it necessary to express in the string description portion of the rule the acceptability of any one of a number of characters, for example any consonant. Of course, this could be expressed in a choice bracket. But in the alphabet of the Thai language, for which TRANS is the primary application, there are 42 consonants. This would require writing a rather long choice bracket each time we want to specify in the SD that a consonant must occur. To avoid this excess we allow the user to define and use abbreviations. The first character of an abbreviation is a slash (/). This may be followed by any number of consecutive alphabetic characters. It must be terminated by a blank, equal sign, comma, asterisk, left parenthesis, or right parenthesis.

When the SD of the rule is processed and a slash occurs, the abbreviation that follows is replaced by the full definition. The program system of which TRANS is a part allows abbreviations and definitions to be entered on-line by means of a command, DEFINE, followed by the abbreviation, an equal sign and the definition, which is to replace every occurrence of the abbreviation in the SD. For example, we might enter the definition for a vowel in English as: DEFINE VW = (A, E, I, O, U). Then when the SD in (62a) is processed each abbreviation is replaced, with the resulting SD as in (62b).*

*Conceptually the SD represented in (62b) in accurate. However, definitions that would be replaced by a long choice bracket are, in practice, replaced by a unique character, which signals the string matching program to enter a subroutine that checks for vowels, consonants, etc.

(62a) . X M *1/VW /VW X =

(62b) . X M *1 (A, E, I, O, U) (A, E, I, O, U) X =

There are no restrictions on the definition and no absolute necessity that it be enclosed within a choice bracket. In fact, if the abbreviation above were to occur inside a choice bracket as in (63a) the resulting SD after replacement would be incorrect because of the superfluous set of parentheses.

(63a) . X (M, /VW) X =

*(63b) . X (M, (A, E, I, O, U)) X =

In a csse such as (63a) it would be desirable to have another abbreviation whose definition did not include parentheses, e.g.

DEFINE VWX = A, E, I, O, U

Then the SD in (63a) could be correctly written as in (64a) with the abbreviation correctly replaced as in (64b).

(64a) . X (M, /VWX) X =

(64b) . X (M, A, E, I, O, U) X =

EXCEPTIONS AND GOTO

The string change portion of a rule may be optionally followed by any one or combination of three types of entries, XOND, COND, and GOTO. Only one GOTO may appear in a rule, and if it does occur, it must be the last entry in the rule. Any number of XOND and COND entries may occur in any order. The XOND, COND and GOTO entries are separated by the colon character. The last entry in this portion of the rule is terminated by a period.

XOND

The XOND entry imposes a restriction or restrictions on a labelled Z variable or a labelled abbreviation. For example, up to this point we would have expected the SD in (65) to match any three-character data string that begins with A and ends with B, the Z variable matching any single character. However, the XOND at the end of SD (65) imposes the further condition on the character labelled *1 (this Z). That further restriction is that this character must not be a C or D.

```
        (65)     . A *1Z *2B  =  *2=0.   XOND *1 CD.
```

Thus, SD (65) will match any three-character data string whose first character is A, whose last character is B, and whose middle character is not C or D.

Let us suppose that /CN were defined as a choice bracket containing all of the 21 letters in the Roman alphabet commonly called consonants. Then SD (66) would match any data string of any length (greater than zero) as long as its first character is a consonant other than H, K, and Q.

```
        (66)     . *1/CN X  =  *1=0.   XOND *1  HKQ.
```

Given the same definition for /CN as stated above, SD (67) will match any data string that ends in two consonants as long as both the second-to-last character is not a B and the last character is not L. Be careful to note that these are two <u>independent</u> conditions. If either is violated a successful match will not occur. Thus, for example, the data strings AMML and AMBM are <u>not</u> matched by SD (67).

```
(67)  .X *1/CN *2/CN  =  *1=*2.  XOND *1 B:XOND *2 L.
```

In the case where it is desired to express the non-independence of these two XOND conditions (for example, the acceptability of AMBM and AMML, but the rejection of AMBL) it is necessary either to use more than one rule or to extend the language by use of a new COND subroutine, discussed below.

COND

COND is a signal for the pattern matching program to invoke a certain subroutine as an additional condition on the normal requirements for a successful match between the SD of a rule and the data string. The particular subroutine to be executed is specified in the field immediately following COND and is represented by a one- or two-digit decimal number. The input parameter to this subroutine is a label (asterisk plus numeral) in the next field.

(68) .A *1Z *2Z = *1=0. COND 5 *2.

Thus, in the operation of the rule in (68), subroutine number 5 will be operated at the time that the character Z labelled *2 is being matched. Let us suppose that the subroutine called by condition 5 demands, for a successful exit, that the labelled character specified must be the same as the immediately previous character. The SD in (68) then will match only data strings containing three characters, where the first character is A and the last two characters may be any character as long as they are identical; for example, AAA, ABB, ACC, ADD, etc.

Rules (69a) and (69b), which are each identical to (68) except for the addition of the XOND entry, would match every data string matched by (68) except ABB and ADD.

(69a) .A *1Z *2Z = *1=0. COND 5 *2: XOND *1 BD.

(69b) .A *1Z *2Z = *1=0. COND 5 *2: XOND *2 BD.

In the case of (69a) and (69b), because of the identity requirement in COND 5, the same result is obtained by imposing the XOND either on *1 or *2. It can thus be seen that the COND capability allows the TRANS language to be greatly extended merely by adding subroutines.

GOTO

The order in which rules are executed is, in general, the order in which rules were input, converted and saved. If three rules were input in the following order A1, A2, A3, then after rule A1 is finished, rule A2 is operated. After A2 is finished, A3 is operated, etc. A repeatable rule is operated until a successful match is not obtained.* Then the next sequential rule is operated.

The GOTO entry provides an alternative to the sequencing of rules described in the paragraph below. The GOTO entry may specify the name of the next rule to be executed in the case where:

(1) the rule successfully matches the data string

(2) the rule fails to match the data string

*Because it is possible with a repeatable rule to get into an interminable loop, the system prevents any rule from operating more than 8 successive times.

One GOTO entry may specify either (1) or (2) or both (1) and (2). The form of a GOTO entry is: GOTO $\underbrace{\underline{\text{rule name}}}_{\text{rule matches}}$ * $\underbrace{\underline{\text{rule name}}}_{\text{rule fails}}$.

The asterisk and at least one rule name must appear in the GOTO entry. The position of the rule name with respect to the asterisk determines the condition under which the rule whose name is mentioned will be executed as the next rule.

If rule A1 contains a GOTO entry as in (70), then rule A6 will operate immediately after A1 if A1 <u>fails</u> to match the data string. But rule A5 will operate immediately after A1 if A1 <u>successfully</u> matches the data string.

(70) RULE A1 string description = string change. GOTO A5 *A6.

In a GOTO entry, the rule whose name appears after the asterisk is operated next in case of a failure to match the data string. The rule whose name appears in front of the asterisk is operated in case of a successful match.

It is not necessary that two names appear in the GOTO entry. Absence of a name after the asterisk means the next sequential rule is to be executed in case of a failure. Absence of a name in front of the asterisk indicates the next sequential rule is to be operated in case of a successful match. If rule A2 is the next sequential rule after A1, then when the rule in (71) is operated, B4 will operate immediately after A1 if the rule successfully matches; but A2, the next sequential rule, will operate if A1 fails to matche the data string.

(71) RULE A1 string description = string change. GOTO B4*.

Again, assume A2 is the next sequential rule after A1. If the rule in (72) fails, then D6 will be operated next. If it is successful A2 is the next rule.

(72) Rule A1 string description = string change. GOTO *D6.

A GOTO entry may <u>not</u> be attached to a repeatable rule. Repeatable rules continue operation until they fail to match. Then the next sequential rule is operated. All non-GOTO rules operate sequentially after failure to match.

The problem discussed in the last paragraph under XOND, page B-22, where it is necessary to accept all strings that end in two consecutive consonants except where the second-to-last consonant is B while the last consonant is L, can now be solved easily with two rules using a GOTO as in (73).

```
(73)    RULE 01   . X *1B L  =  *1=*1.GØTØ  03*.
        RULE 02   . X *1/CN *2/CN  =  *1=*2.
        RULE 03   -
```

The string change in RULE 01 has no effect. It is there merely to fulfill the requirement that all rules have a string change.

One further point of difficulty can best be clarified in the following example. Using the definition for /VWX provided on page B-20, SD(74) <u>will</u> match the string in (75). At first the 'A' is accounted for by /VWX. However, the XOND prevents *1Z from matching a 'B'. The matching algorithm therefore backs up to 'A', considers it as optional, and is therefore able to account for it using *1Z, leaving the X variable to account for 'BC'. After the operation of this rule, (75) is changed to appear as in (76).

```
(74)    (/VWX, %) *1Z X=  *1=0. XOND *1 B.
(75)    ABC
(76)    BC
```

APPENDIX C

TRACTS: THAI/ROMAN COMPUTERIZED TRANSLITERATION SYSTEM

This appendix presents the results of a study undertaken to determine the modifications to TRACTS, THAI/ROMAN COMPUTERIZED TRANSLITERATION SYSTEM, necessary to produce a transliteration that will indicate fully the pronunciation of a Thai word given its spelling. Included are a description of the form of this expanded output as well as the new rules and rule modifications and deletions that would be necessary to effect this output.

INTRODUCTION

A set of computer programs, "TRACTS: Thai/Roman Computerized Transliteration System," has been written for operation on an IBM 1800 computer. This system accepts as input words that are typed in Thai characters. It produces as output a standardized transliteration (transcription) of the Thai input word in Roman characters. For example, if the word ปริศนา were input, the system would output PRIT/SA/NA. The standards adhered to are those of the Royal Institute of Thailand as set forth in their publication "Romanization Guide for Thai Script," April 1968, under the heading "The General System."

In order to utilize these standards the user must know the pronunciation of the Thai word to be transliterated. The primary problem of TRACTS, then, was to deduce the pronunciation of a Thai word from its spelling. This was accomplished and in the final weeks of testing the program was producing output with 96% accuracy.

Although the transliterated output is based on the pronunciation of the Thai word, it does not fully indicate the word's pronunciation. Certain phonemes of Thai are, following the rules of the Royal Institute, suppressed. In other cases, the distinction between pairs of phonemes is lost by having both members represented by the same Roman letter in the transliterated output. As an example consider (1) through (5) below. Each is a separate word in Thai, different in meaning and pronunciation (indicated in parentheses), and yet each, according to the General System of the Royal Institute, is transliterated identically as "FAN."

(1) ฟัน (fan) - to sever

(2) ฟั่น (fân) - to twist together

(3) ฟั้น (fán) - to massage vigorously

(4) ฝัน (fǎn) - to dream

(5) ฝาน (fǎan) - to slice using a horizontal motion

While the General System of the Royal Institute is appropriate for the uses for which it was intended--e.g., international maps--a transliteration that indicates a full native speaker, is desirable for other purposes, such as educational applications. To this end we have undertaken this study and produced the following catalogue of changes and additions that would have to be made to TRACTS to produce the full phonemic output. These modifications have been tested to the extent possible on paper but have not been implemented in the computer system.

The two major types of phonemes now suppressed are syllable tone and vowel length. These are discussed separately in the following pages and followed by a section that covers all of the other changes necessary to produce a phonemic output.

Syllable Tone

The relative pitch or tone at which a syllable or syllables of an utterance are produced in Thai is used by speakers to indicate different words of the language. Tone is as important for distinguishing words in Thai as is the difference between /m/ and /n/ or any of the other phonemes of the language.

In Thai five tones are used to distinguish among words (that is, five tones are phonemic): mid-tone, low-tone, falling-tone, high-tone, and rising-tone. Table C1 shows five words that are distinguished by tone only. In the General System of transliteration these would all be represented as "NA." Because the symbols for tone that are suggested by the International Phonetic Association, and the widely accepted symbols for Thai tone phonemes used by Dr. Mary Haas are not available on standard data processing equipment or on the Shinko teleprinter used for input/output in TRACTS, we have had to use other symbols. These we use are the numbers 1, 2, 3, and 4 in front of the vowel of [illegible] order to represent low, low, falling, high, [illegible] is unmarked.

In order to output these indicators of syllable tone, fourteen new rules were written. These rules make use of nine new abbreviations, which are defined in Table C-3, page C-18.

Vowel Length

The relative duration of a vowel is phonemic in Thai; words that are otherwise identical in sound may be distinguished by the relative length of their vowels. Table C-2 gives minimal pairs for most of the vowels in Thai (no minimal pairs have been found for the long and short diphthong) and the new representations used to differentiate between long and short vowels.

TABLE C-1

MINIMAL PAIRS IN THAI, WHERE SYLLABLE TONE IS THE ONLY DISTINGUISHING FEATURE.

Thai Word and Meaning	Pronunciation (Haas Symbols)	Transliteration, General System	Transliteration, Full Phonemic
นา	naa	NA	NAA
field			
หน่า *	nàa	NA	N1AA
custard apple			
หน้า	nâa	NA	N2AA
face			
น้า	náa	NA	N3AA
maternal uncle or aunt			
หนา	nǎa	NA	N4AA
to be thick			

*last syllable in the word น้อยหน่า

TABLE C-2

MINIMAL PAIRS IN THAI, WHERE VOWEL LENGTH IS THE ONLY DISTINGUISHING FEATURE.

Thai Word and Meaning		Pronunciation (Haas Symbols)	Transliteration, General System	Transliteration, Full Phonemic
วัน	day	wan	WAN	WAN
วาน	to ask	waan	WAN	WAAN
จิบ	to sip	cìb	CHIP	C1IP
จีบ	to pleat	cìib	CHIP	C1IIP
อึด	to suppress	ʔỳd	UT	ʔ1YT
อืด	swollen	ʔỳyd	UT	ʔ1YYT
ยุง	mosquito	juŋ	YUNG	JUNG
ยูง	peafowl	juuŋ	YUNG	JUUNG
เอ็น	tendon	ʔen	EN	ʔEN
เอน	to lean	ʔeen	EN	ʔEEN
แกะ	sheep	kɛ̀ʔ	KAE	K1AE
แก	to be old	kɛ̀ɛ	KAE	K1AEAE
ปน	to mix	pon	PON	PON
โปน	to bulge	poon	PON	POON
เกาะ	island	kɔ̀ʔ	KO	K1)
ก่อ	to build	kɔ̀ɔ	KO	K1))
เจอะ	to meet	cə̀ʔ	CHOE	C1OE
เจ่อ	to swell	cə̀ə	CHOE	C1OECE

Other Changes in the Representation of Phonemes

In addition to representing vowel length and syllable tone, which were suppressed in the Royal Institute's General System, we also make the following distinctions that are necessary for a full phonemic output but that were not made by the Royal Institute:

(1) C, CH

The General System represents both จ and the group ฉ, ช, ฌ by the characters CH when they occur as the initial syllable. However, the sound represented by จ, /c/, an unaspirated voiceless palatal stop, is phonemic in Thai and contrasts with the syllable initial sound of the other three, /ch/, which is aspirated. An example of a minimal pair would be the words จุก /cùg/, (bottle stopper) and ฉุก /chùg/, (to happen suddenly).

(2) O,)

The mid back rounded vowels /o/, /oo/ are not distinguished from the low back rounded vowels /ɔ/, /ɔɔ/, being represented in the General System as "O". We use the symbols "O", "OO" and ")", "))", respectively.

(3) Y, U

The high central unrounded vowels /y/, /yy/ are represented as "U" in the General System as are the high back rounded vowels /u/, /uu/. We represent the former as "Y", "YY" and the latter as "U", "UU".

(4) ʔ

The glottal stop phoneme /ʔ/, which is signalled by Thai character "อ" syllable initial, is omitted by the Royal Institute General System. We represent it in all of its occurrences as "ʔ".

(5) Y, J

The palatal semivowel sonorant /j/ is represented in the General System as "Y". Since the authors, following the conventions used by Dr. Mary Haas, use "Y" for the high central unrounded vowel, we must use a different symbol for the semivowel. We follow Dr. Haas again, using the symbol "J".

<u>System Change to Provide Full Phonemic Output</u>

TRANS RULES

In order to effect tne insertion of the symbols to represent syllable tone it was necessary to write fourteen new TRANS rules. These rules would be added to the TRANS rules described in SDC document TM-(L)-4681/000/00, TRACTS: Thai/Roman Computerized Transliteration System, 1 February 1971. These new rules are to be added immediately after rule Iน , page 70 of that document. The new TRANS rules are to be found in Table C-4, page C-19. These rules make use of nine new abbreviations, which are defined in Table C-3, page C-18. Annotations and examples for the new rules follow.

RULE VA

At this point in the operation of the system, tone marks can occur either before a mid-positioned vowel, e.g. า , or after a super-positioned, sub-positioned, or preposed vowel. In order to normalize the occurrence of tone marks for the facility of the other fourteen rules, rule VA exchanges the position of the tone marks (่ , ้ , ๊ , ๋) with an immediately preceding super-positioned, sub-positioned, or preposed vowel so that in all cases tone marks will precede the vowel, e.g.:

	จ ั ่ ว	becomes	จ ่ ั ว.
	ข ิ ่ ม	becomes	ข ่ ิ ม.
	ส ุ ้	becomes	ส ้ ุ
	น ไ ้ ม	becomes	น ้ ไ ม
but	ป ้ า ย	remains	ป ้ า ย

RULE VB

A syllable initiated by a middle consonant that has a short vowel (note that this includes the implicit vowel "O" but not implicit vowel "A") and that has a stop syllable final is pronounced with low tone. Thus, the symbol "1" is inserted before the vowel, e.g.:

ก ั ก	becomes	ก 1 ั ก
ก ว ั ด	becomes	ก ว 1 ั ด

RULE VC

An open syllable (syllable without a final consonant) that is initiated by a high consonant and that contains a long vowel is pronounced with rising tone. Thus:

ฉ า	becomes	ฉ 4 า
ห ม ู	becomes	ห ม 4 ู

RULE VD

A syllable that is initiated by a high consonant and that contains an explicit vowel symbol (or symbols, if it is a complex vowel) and that is terminated by a sonorant is pronounced with rising tone. Thus:

ห า น	becomes	ห 4 า น
ผ ั น	becomes	ผ 4 ั น
ห ว ั ง	becomes	ห ว 4 ั ง
ส เ ี ย ง	becomes	ส 4 เ ี ย ง

RULE VE

A syllable that is initiated by a high consonant and terminated by a short vowel is pronounced with low tone. Thus:

ส ุ	becomes	ส 1 ุ
ข ะ	becomes	ข 1 ะ

RULE VF

A syllable initiated by a high consonant, terminated by a stop and containing an explicit vowel character or complex vowel characters is pronounced with low tone. Thus:

ข ุ ก	becomes	ข1ุก
ขว ั ก	becomes	ขว1ัก

RULE VG

An open syllable that is initiated by either of the two types of low consonants and that contains a long vowel or the character อ functioning as a vowel plus the tone character " ' ", mái̯ʔèeg, is pronounced with falling tone. Thus:

ค ่ า	becomes	ค2า
คร ่ ู	becomes	คร2ู

RULE VH

A syllable that contains the tone mark " ' ", mái̯ʔèeg, is initiated by a low consonant of either type, ends in a sonorant, and contains an explicit vowel symbol or " ว " or " อ " functioning as a vowel is pronounced with falling tone. Thus:

คล ่ ุ ม	becomes	คล2ุม
ง ่ าม	becomes	ง2าม

RULE VI

A syllable that is initiated by a low consonant of either type, contains májthoo, " ้ ", and is terminated by a long vowel or " อ " functioning as a vowel is pronounced with rising tone. Thus:

ช ้ ำ	becomes	ช 3 ำ
น ้ า	becomes	น 3 า

RULE VJ

A syllable that is initiated by a low consonant of either type, contains the symbol májthoo, " ้ ", a vowel, a complex vowel, or the characters " ว " or " อ " functioning as a vowel, and is terminated by a sonorant, is pronounced with high tone. Thus:

น ้ ิ ว	becomes	น 3 ิ ว
ม ้ า ม	becomes	ม 3 า ม

RULE VK

A syllable that is initiated by a low consonant of either type, contains a long vowel or " ว " or " อ " functioning as a vowel, is terminated by a stop, and contains no tone mark is pronounced with falling tone. Thus:

ม า ก	becomes	ม 2 า ก
พ ล า ด	becomes	พ ล 2 า ด

RULE VL

A syllable that is initiated by a low consonant of either type, contains a short vowel, and is terminated by a stop and contains no tone mark is pronounced with high tone. Thus:

ม ั ก	becomes	ม3 ั ก
พ ล ิ ก	becomes	พ ล3 ิ ก

RULE VM

A syllable initiated by a low consonant of either type, that contains a short vowel, májʔèeg, "่", and is terminated by a stop is pronounced with falling tone. Thus:

ล ่ ั ก	becomes	ล 2 ั ก
ล ่ ุ บ	becomes	ล 2 ุ บ

RULE VN

A syllable initiated by a low consonant of either type, that contains májthoo, " ้ ", a long vowel or "ว " or "อ " functioning as a vowel, and that terminates in a stop is pronounced with high tone. Thus:

ฟ ้ า ค	becomes	ฟ 3า ค
ว ้ ี ค	becomes	ว 3 ี ค

RULES TN and HH

In addition to the fourteen new TRANS rules above, it will be necessary to remove rule TN in order to preserve the tone marks (่ , ้ , ๊ , ๋) in words that were not affected by rules VA through VN. These will then be converted to the symbols "1, 2, 3, 4", respectively, by the new Romanization rewrite rules. Thus, for example, ก่ำ is not affected by the new TRANS tone rules. However, the tone mark, májʔèeg "่" remains; it is subsequently converted to "1" by a rewrite rule in order to correctly indicate that the word is pronounced with low tone, ก 1ำ .

Further, rule HH will have to be changed in order to permit the rewrite rules to output a syllable initial glottal stop symbol "ʔ". In TRACTS, syllable initial "อ" in an IVS is changed to "O" so that there will be no vowel-less syllables. Rule HH will be changed as below to retain the อ , which is changed to "ʔ" by a rewrite rule, and to insert the implicit vowel "O". Thus, "อด ", by the rule below will become อ O ด and will eventually be output as ?1OD.

RULE HH R X & *1 อ *2/CN& X = *1 + *1 + Ø.

The Exception Lists

In the original version of TRACTS, when a full-word exception was found, execution of the TRANS rules was terminated and the unaltered replacement sequence was immediately Romanized by the rewrite rules. With the addition of the new TRANS rules VA through VN after rule I น, it now becomes necessary to evaluate the replacement sequence for tone also. Accordingly, full-word exceptions will begin operation of the TRANS rules at VA. Of course, all other sequences, partial word exceptions as well as nonexceptions, will be evaluated by these new rules.

Certain changes will have to be made to the exception lists themselves since they were created to be output according to the Royal Institute rules. These changes, however, are minimal. They consist merely of changing certain of the replacement sequences in the exception lists. These are shown in Table C-5, page C-20. The left-hand column of this table represents the replacement sequence to be searched for. If it is found, it is changed to appear as indicated in the corresponding right-hand column. For example, on page 79 of the TRACTS document (TM-(L)-4681/000/00) we find the exception word ระเหย with the replacement sequence & ระ & HOEI &. By the first rule in Table C-5, this replacement sequence is changed to & ระ & H4OEOEJ.

As another example, consider entry (13) in Table C-5. According to this, we must change the character "D", wherever it appears in a replacement sequence, to the character ฑ . On page 85 of TM-(L)-4681/000/00 we find the exception word บัณฑุ , whose correct pronunciation is BAND1U. The replacement sequence for this word is บัณ & Dุ. However, the D is changed to ฑ by rule 13 of Table C-5, so that tone rule Vb of Table C-4 can insert the tone "1".

<u>Rewrite Rules</u>

The following changes necessary to produce the phonemic output are accomplished by the revised version of the Romanization rewrite rules in Table C-6, pages C-21 through C-24.

(1) Differentiation between long and short vowels.

(2) Differentiation between /c/, /ch/; /o/, /ɔ/; /y/, /u/; /y/, /j/.

(3) Conversion of the tone marks (่, ้, ๊, ๋) not handled by rules VA - VN.

Table C-6 is interpreted just as Table 4 on page 18 of TM-(L)-4681/000/00. It presents a complete version of the rewrite rules for vowels and, thus, corresponds to rules 1-40 of the TRACTS rewrite rules. Rules 41-79 remain the same for our purposes. They are not listed here except for the changes in the four rules 44, 45, 50, and 54.

In Table C-6, where a rule is numbered, the number refers to the comparable rule in TRACTS. Thus, for example, both rule 5 in TRACTS and rule 5 of this document rewrite ัว. Rules that are not numbered are new rules. It should be noted that rules 1 and 2 of TRACTS are omitted from this new version.

CONCLUSION

The purpose of this study was to determine whether the existing computer-operated transliteration system, TRACTS, could be expanded or modified to produce an output that would indicate fully the pronunciation of a Thai word where the input to the system was the spelling of that word in Thai script. We believe that the changes to TRACTS that we have detailed will, with some modification, provide the desired input. Those modifications can only be determined when the changes are actually made to the computer programs and tested by being run on a wide variety of data.

Definition of Terms

Complex Vowel:	A vowel that, in its written representation, is represented by more than one character; for example, the diphthong /ua/ in the word หัว /hua/ is represented by the characters ั ว.
IVS:	Implicit vowel syllable; each syllable in the Thai language contains a spoken vowel. However, some syllables in their representation contain no explicit vowel character. Such a syllable is referred to as an implicit vowel syllable.
Minimal Pair:	Two utterances that differ from each other by only one phoneme, e.g., in English the words rap and rat.
Phoneme:	A member of the set of the smallest units of speech that serve to distinguish one utterance from another in a language or dialect.
Replacement Sequence:	Before being altered by TRANS rules and Romanization rewrite rules, each Thai input word is compared with a list of exception words. If a match is found, the Thai word or a portion of it is changed by having a replacement sequence of characters substituted for that portion of the word that matched the exception word or exception sequence.

TABLE C-3

NEW ABBREVIATIONS AND DEFINITIONS TO BE ADDED TO TRACTS. THESE ABBREVIATIONS ARE REFERENCED IN THE NEW TONE RULES VA - VN.

Name	Abbreviation		Definition
stop	/STOP	=	(ก, ข, ค, ฆ, จ, ฉ, ช, ฎ, ฏ, ฐ, ฑ, ฒ, ด, ต, ถ, ท, ธ, บ, ป, พ, ฟ, ภ, ศ, ษ, ส, P, T, K)
sonorant	/SONOR	=	(ง, ญ, ณ, น, ม, ย, ร, ล, ว, ฬ, M, N, NG, W, J)
muddle consonant	/ MCN	=	(ก, จ, ฎ, ฏ, ด, ต, บ, ป, อ)
high consonant	/ HCN	=	(ข, ฃ, ฉ, ฐ, ถ, ผ, ฝ, ศ, ษ, ส, ห)
low consonant Type 1	/LCN1	=	(ค, ฅ, ฆ, ช, ซ, ฌ, ฑ, ฒ, ท, ธ, พ, ฟ, ภ, ฮ)
low consonant Type 2	/LCN2	=	(ง, ญ, ณ, น, ม, ย, ร, ล, ว, ฬ)
short vowel	/SVW	=	(ะ, ั, ัวะ, ิ, ึ, ุ, เะ, เ็, เาะ, เอะ, เียะ, เือะ, แะ แ็, โะ, ือ, O)
long vowel	/LVW	=	(ัว, เอ, เีย, เือ, เิ, ือ, เา, า, ี, ื, ู, เ, แ, โ, ใ, ไ, ำ, UA, UAJ,)),))J, OO)
vowel, complex vowel	/VCVW	=	(/LVW, /SVW)

TABLE C-4

TRANS RULES VA - VN TO BE ADDED AFTER RULE Iแ IN TRACTS TO OUTPUT SYMBOLS FOR SYLLABLE TONE.

```
RULE VA R X *1 ( ั, ิ, ี, ึ, ื, ุ, ู, เ, แ, โ) *2 ( ่, ้, ๊, ๋)X
            =*1 = *2+*1: *2 = 0.

RULE VB R X &/MCN ( ร, ล, ว, % ) *1/SVW  (/STØP, %) &X =  *1 = 1+*1.

RULE VC R X &/HCN (/CN, %) *1/LVW &X = *1 = 4+*1.

RULE VD R X &/HCN (/CN, %) *1(/VCVW, ว, อ ) /SONOR & X = *1 = 4+*1.

RULE VE R X &/HCN (/CN, %) *1/SVW & X = *1 = 1+*1.

RULE VF R X &/HCN (/CN, %) *1(/VCVW, ว, อ ) /STØP & X = *1 = 1+*1.

                                                  ่
RULE VG R X &(/LCN1, /LCN2) ( ร, ล, ว, %) *1 (/LVW, อ ) & X = *1 = 2.

                                                  ่
RULE VH R X &(/LCN1, /LCN2) ( ร, ล, ว, %) *1 (/VC'W, ว, อ ) /SONOR & X = *1 = 2.

                                                  ้
RULE VI R X &(/LCN1, /LCN2) ( ร, ล, ว, %) *1 (/LVW    , อ) & X = *1 = 3.

                                                  ้
RULE VJ R X &(/LCN1, /LCN2) ( ร, ล, ว, %) *1 (/VCVW, ว, อ) /SONOR & X = *1 = 3.

RULE VK R X &(/LCN1, /LCN2) ( ร, ล, ว, %) *1 (/LVW, ว, อ) /STØP & X = *1 = 2+*1.

RULE VL R X &(/LCN1, /LCN2) ( ร, ล, ว, %) *1/SVW /STØP & X = *1 = 3+*1.

                                                  ่
RULE VM R X &(/LCN1, /LCN2) ( ร, ล, ว, %) *1 /SVW /STØP & X = *1 = 2.

                                                  ้
RULE VN R X &(/LCN1, /LCN2) ( ร, ล, ว, %) *1 (/LVN, ว, อ) /STØP & X = *1 = 3.
```

TABLE C-5

REPLACEMENT SEQUENCES IN THE TRACTS EXCEPTION LISTS THAT NEED TO BE CHANGED TO GIVE PHONEMIC OUTPUT.

	Replacement Sequence or Portion of Replacement Sequence in TRACTS		Change in Sequence
(1)	HOEI	=	H4OEI
(2)	OEI	=	OEOEJ
(3)	RU '	=	ร ้
(4)	O,	=	))
(5)	TRI	=	ต ร ้
(6)	ตAO	=	ต า W
(7)	หAO	=	ห า W
(8)	ROE ก	=	เ ริ ก
(9)	ROEK	=	เ ริ ก
(10)	THOEN	=	เ ทิ น
(11)	THOEM	=	เ ทิ ม
(12)	THOET	=	เ ทิ ต
(13)	D	=	ต
(14)	& ท O ต & THRO ' &	=	& ท Oต & THR โ้ &
(15)	& THRO &	=	& THR โ &

TABLE C-6

REVISED VERSION OF ROMANIZATION REWRITE RULES. THE SEQUENCE IN THE LEFT COLUMN IS REWRITTEN TO APPEAR AS ON THE RIGHT. THE NUMBERS IN PARENTHESES ARE CROSS-REFERENCES TO RULES OF THE SAME NUMBER IN THE ORIGINAL TRACTS.

(3)	/CN (่ , ้ , ๊ , ๋ , %) ว ย	=	/CN UAJ
	/CN (่ , ้ , ๊ , ๋ , %) ว /CN	=	UA
(4)	ั ว ะ	=	UAʔ
	ั ว	=	UA
(5)	ั ย	=	AJ
(6)	า ย	=	AAJ
(7)	า ว	=	AAW
(8)	ิ ว	=	IW
(9)	ุ ย	=	UJ
(10)	ู ย	=	UUJ
	่เ/SØNØR &	=	่E/SØNØR &
	๋เ/SØNØR &	=	๋E/SØNØR &
(11)	เ ว	=	EEW
	เ ็	=	E
	เ ็ว	=	EW
(12)	เ ย	=	ØEØEJ
(13)	เ อ ะ	=	ØE
	เ อ	=	ØEØE
(14)	เ ะ	=	E
(15)	เ าะ	=	)
(16)	เ า	=	AW

TABLE C-6
(Continued)

(17)	เ ิ	=	ØEØE
(18)	เี ย ว	=	IAW
(19)	เี ย ะ	=	IA?
	เี ย	=	IA
(20)	เื อ ย	=	YAJ
(21)	/CN (่ , ้ , ๊ , ๋ , %) อ ย	=	/CN (่ , ้ , ๊ , ๋ , %)))J
	/CN (่ , ้ , ๊ , ๋ , %) อ /CN	=	/CN (่ , ้ , ๊ , ๋ , %)))
	/CN ื อ ย	=	/CN)J
	/CN ื อ /CN	=	/CN) /CN
(22)	เื อ ะ	=	YA?
	เื อ	=	YA
	่ แ /SØNØR &	=	่ AE /SØNØR &
	๋ แ /SØNØR &	=	๋ AE /SØNØR &
(23)	แ ว	=	AEAEW
(24)	แ ะ	=	AE
	แ ็	=	AE
(25)	โ ย	=	ØØJ
(26)	โ ะ	=	Ø
(27)	(ก , ต , ท , ป , ศA &, สA &) ฤ	=	(ก , ต , ท , ป , ศA &, สA &) RI
(28)	& ฤ (ท , ษ)	=	& RIT
(29)	ฤา	=	RYY
	ฤ	=	RY

TABLE C-6
(Continued)

(30)	ฦๅ	=	LYY
	ฦ	=	LY
(31)	(ะ , ั)	=	A
	า	=	AA
(32)	ำ	=	AM
(33)	ิ	=	I
	ี	=	II
(34)	ึ	=	Y
	ุ	=	U
	ู	=	UU
(35)	ือ	=	YY
	ื	=	YY
(36)	เ	=	EE
(37)	แ	=	AEAE
(38)	โ	=	ØØ
(39)	/CN (่, ้, ๊, ๋, %)	=	/CN (่, ้, ๊, ๋, %)))
(40)	(ใ, ไ)	=	AJ

TABLE C-6
(Continued)

new or changed consonant and tone rewrite rules

(44)	& อ	=	&?
(45)	& ย	=	&J
(50)	& จ	=	&C
(54)	& ญ	=	&J

new rules to be placed after TRACTS rule 79

่	=	1
้	=	2
๊	=	3
๋	=	4

Erratum

An additional TRANS rule for correct tone representation should be inserted in the list in Table C-4 after rule VB, as follows:

```
RULE  VØ  R  X  & /MCN ( ร , ด, ว, % )  *1 (LVW, ว, บ ) STOP & X = *1= 1+*1.
```

The effect of the combination of the two rules VB and VØ will be that syllables with an explicit vowel initiated by a middle consonant and terminating in either a short vowel or a stop will be pronounced with low tone.

APPENDIX D

SDC-SUPPLIED HARDWARE

The following pieces of equipment were designed, manufactured, and tested by SDC to provide interface capability between the IBM 1800, the Shinko Control Box, and the Thai Telex Network:

Box #1 IBM to Shinko only
#2 IBM to Shinko only
#3 IBM to Shinko or Thai Telex
#4 IBM to Thai Telex only (spare)
#5 IBM to Shinko only (spare).

Cables with connectors interfaced the Shinko Control Boxes and the SDC buffers.

Boxes numbered 1 through 5 have the following characteristics described below.

SDC BOXES #1, 2, AND 5 FOR INTERFACE OF IBM 1800 TO SHINKO CONTROL BOX

Inputs

1. Power; 220 VAC, 50 cycle.

2. EAI connector (EAI voltage levels are as follows):

   $+3.0V < MARK < +20.0V$

   $-20.0V < SPACE < -3.0V$.

   NOTE: With no input signal, logic level at input is negative (space).

3. Three-wire white connector plug.

   a. SEND (color code: green)

      No signal; +0.5 to +0.8V

      Signal; -0.5 to -0.8V.

   b. GROUND RETURN (color code: blue).

Outputs

1. EAI connector (see 2 above).

   NOTE: With no output signal to 1800, output level is negative (space).

2. Three-wire white connector plug (see 3 above).

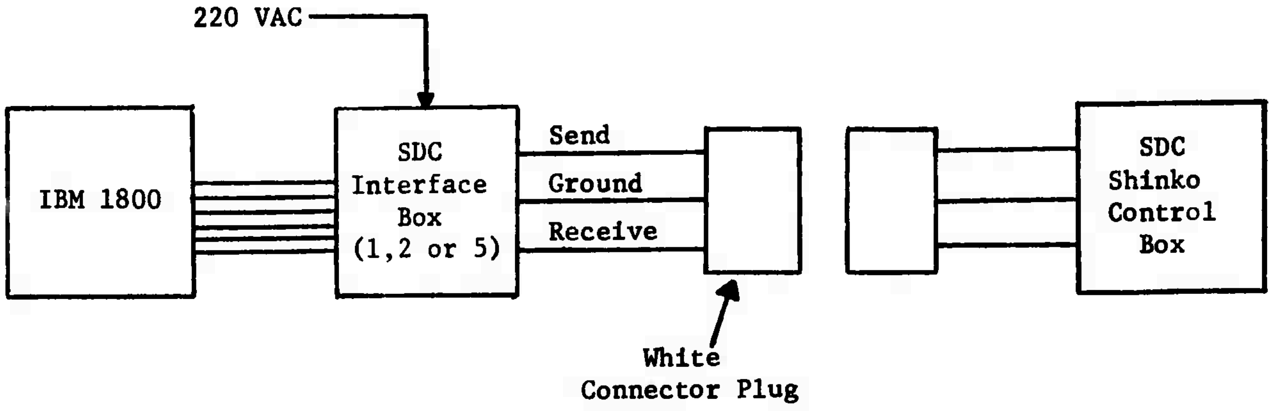
   a. RECEIVE (color code: violet)

      No signal: more than -20V (negative)

      Signal; less than +20V (positive).

   b. GROUND RETURN (same wire as for input ground return).

The system block diagram is as follows:



Circuit Description

The SDC interface box contains the following circuits:

1. Power Supply (± 20V).
2. Inverter Circuit (mounted on coupler card).
3. The repeater circuit (2) mounted on repeater cards A and B. Repeater circuits (relay drivers) include a -30 volt supply.
4. Power-on light driver circuit.

In addition, the 220 VAC to 110 VAC transformer is protected by a series fuse.

Q1 on each repeater card is protected by a base-emitter diode.

The ±20 volt supply on the coupler card supplies power to the inverter and the power-on light and also line current to the Shinko control box.

<u>Special Instructions</u>

1. Before turning on the power to the SDC interface box it should be verified that the two repeater cards inside the box are properly connected.

2. If any alterations are made or parts exchanged due to component failures, the 220 VAC should be disconnected.

<u>SDC BOX #3 FOR INTERFACE OF IBM 1800 TO EITHER THE SHINKO CONTROL BOX OR THAI TELEX</u>

<u>Inputs</u>

1. Power; 220 VAC, 50 cycle.

2. EAI connector from IBM 1800.

3. Three-wire <u>white</u> connector plug from SDC Shinko. (For input levels, see specifications for SDC black box without a switch.)

4. Two-wire Cannon connector from Thai Telex.

<u>Outputs</u>

1. EIA connector from IBM 1800 (same as 2 above).

2. Three-wire <u>white</u> connector plug from SDC Shinko. (For output levels, see specifications for SDC interface box without a switch.)

3. Two-wire Cannon connector from Thai Telex (same as 4 above).

The system block diagram is as follows:

NOTE: With the switch in the "GPO TELEX" position, the interface box provides an interface between the 1800 and the Thai Telex Network.

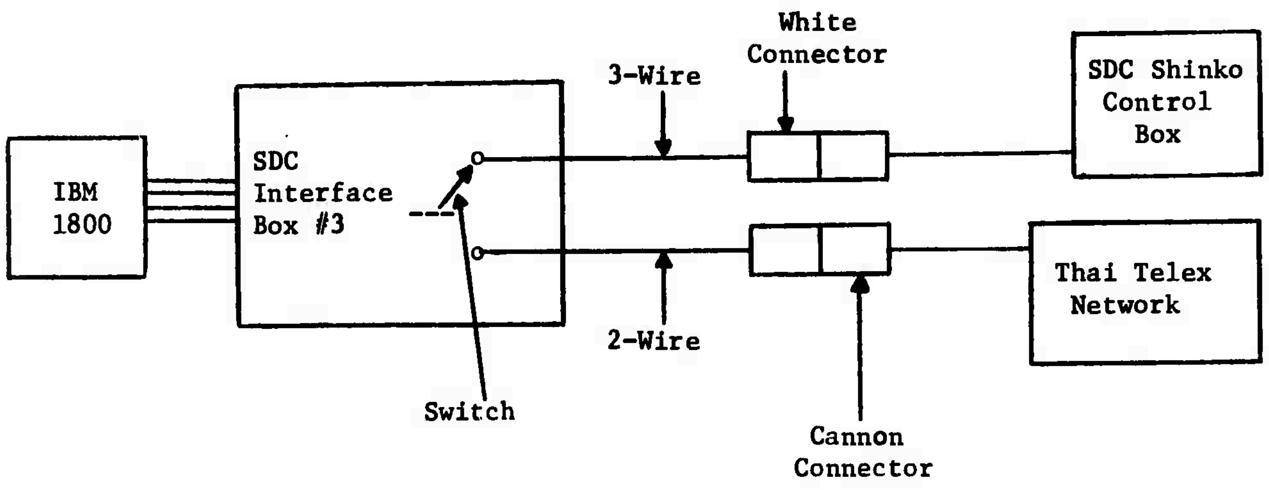
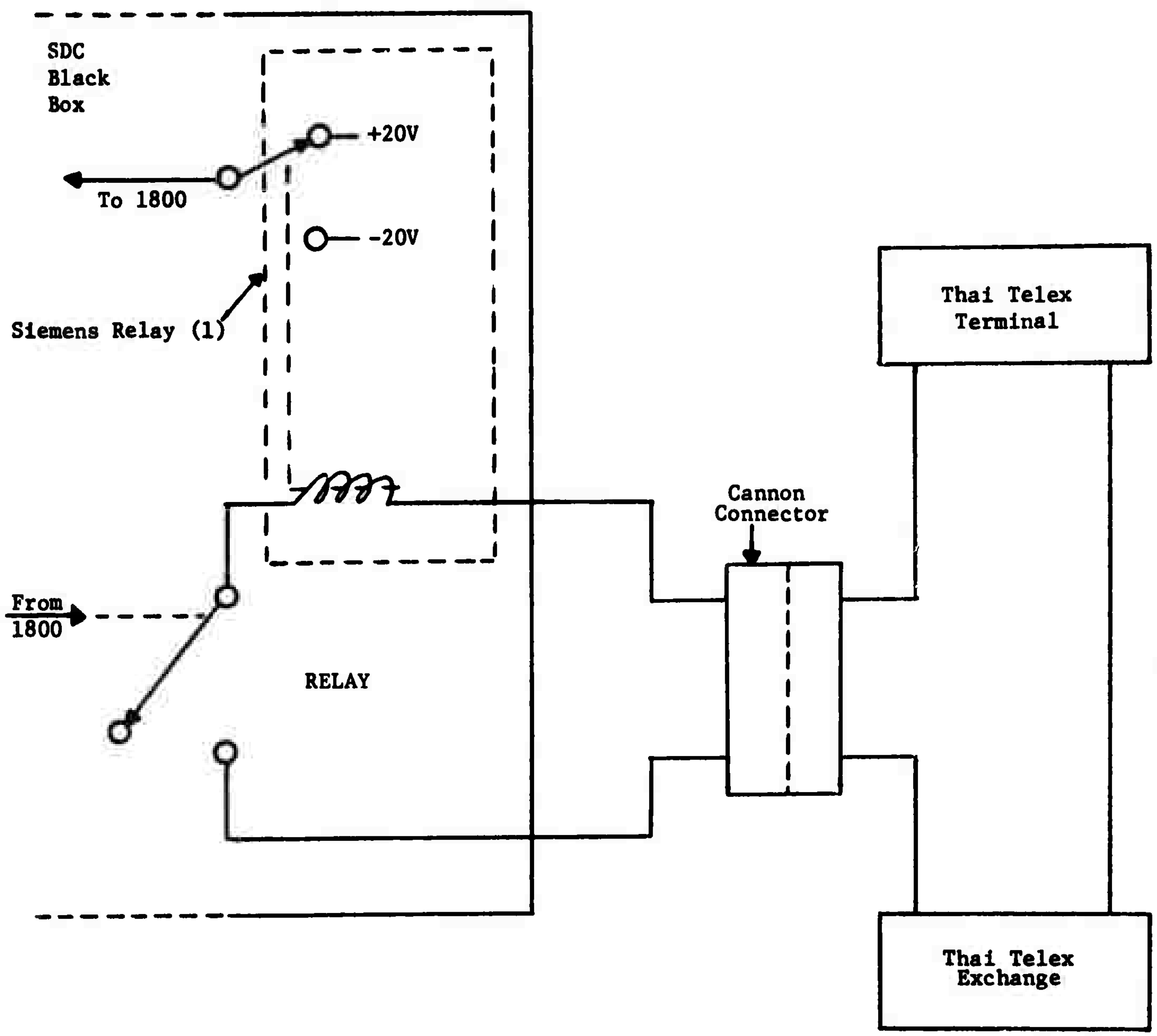


Diagram of 2-Wire Circuit



SDC BOX #4 FOR INTERFACE OF IBM 1800 TO THAI TELEX

Inputs

Same as Boxes 1, 2, 3, and 5.

Outputs

1. EIA connector from IBM 1800.
2. Two-wire Cannon connector from Thai Telex (same as Box #3).

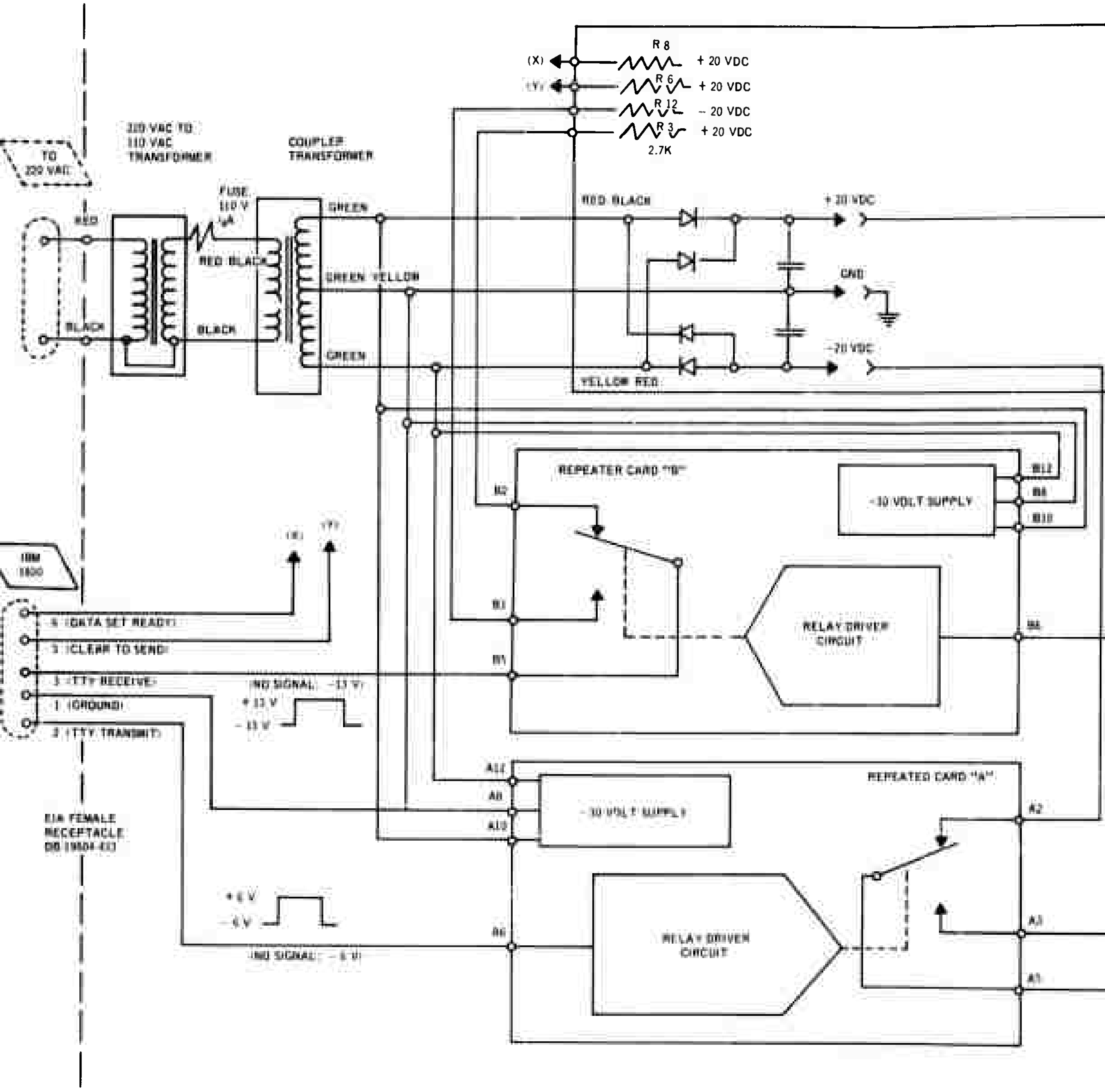
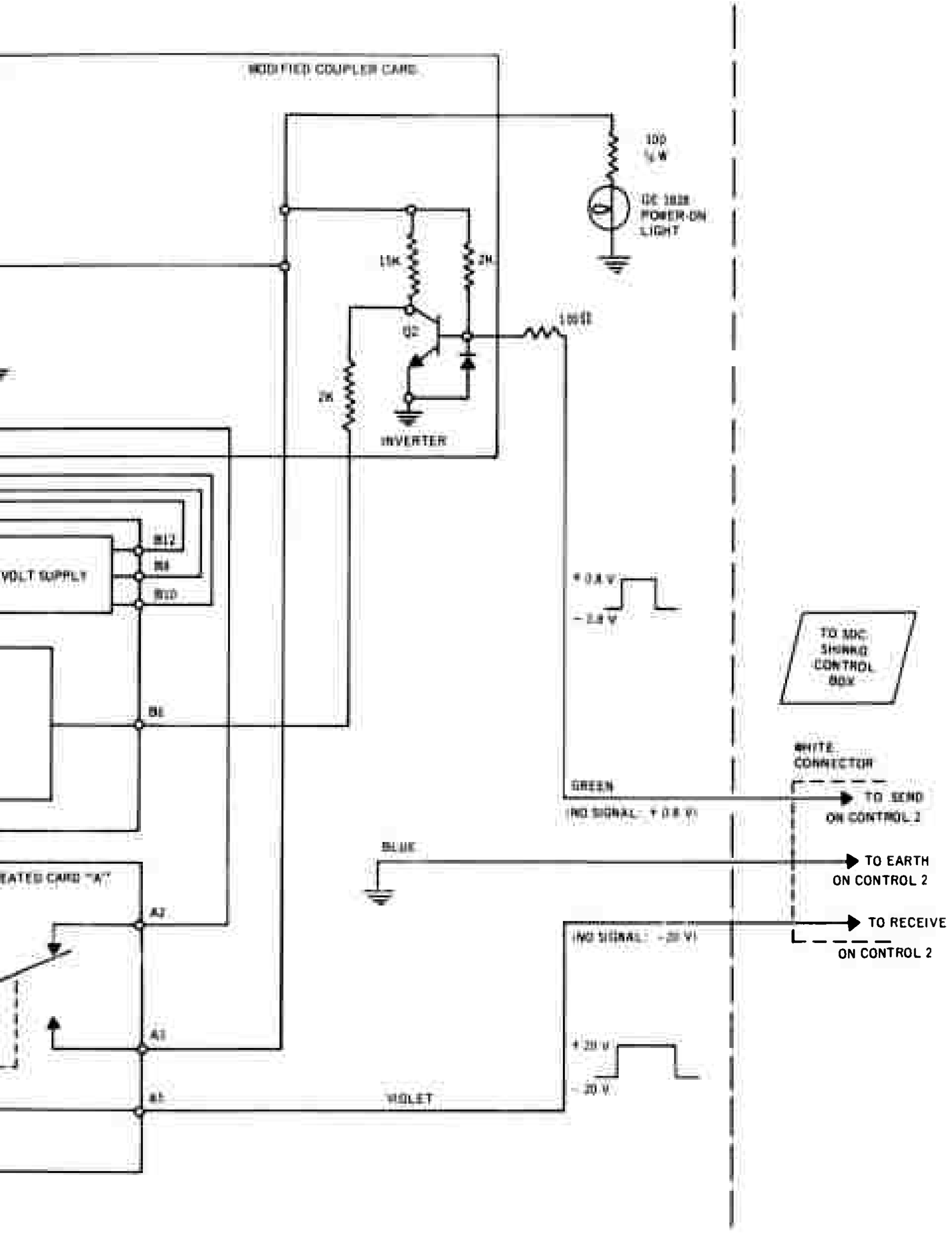


Figure D-1. Schematic SDC Interface Box (#1, 2, and 5), 1800 to SDC Shinko Interface


MODIFIED COUPLER CARD
100
¼ W
GE 1828
POWER-ON
LIGHT
15K
2K
Q2
100Ω
2K
INVERTER
VOLT SUPPLY
B12
B8
B10
B6
+ 0.8 V
– 0.8 V
TO SDC
SHAKO
CONTROL
BOX
WHITE
CONNECTOR
GREEN
(NO SIGNAL: + 0.8 V)
TO SEND
ON CONTROL 2
BLUE
TO EARTH
ON CONTROL 2
EATED CARD "A"
A2
(NO SIGNAL: – 20 V)
TO RECEIVE
ON CONTROL 2
A3
+ 20 V
– 20 V
A1
VIOLET

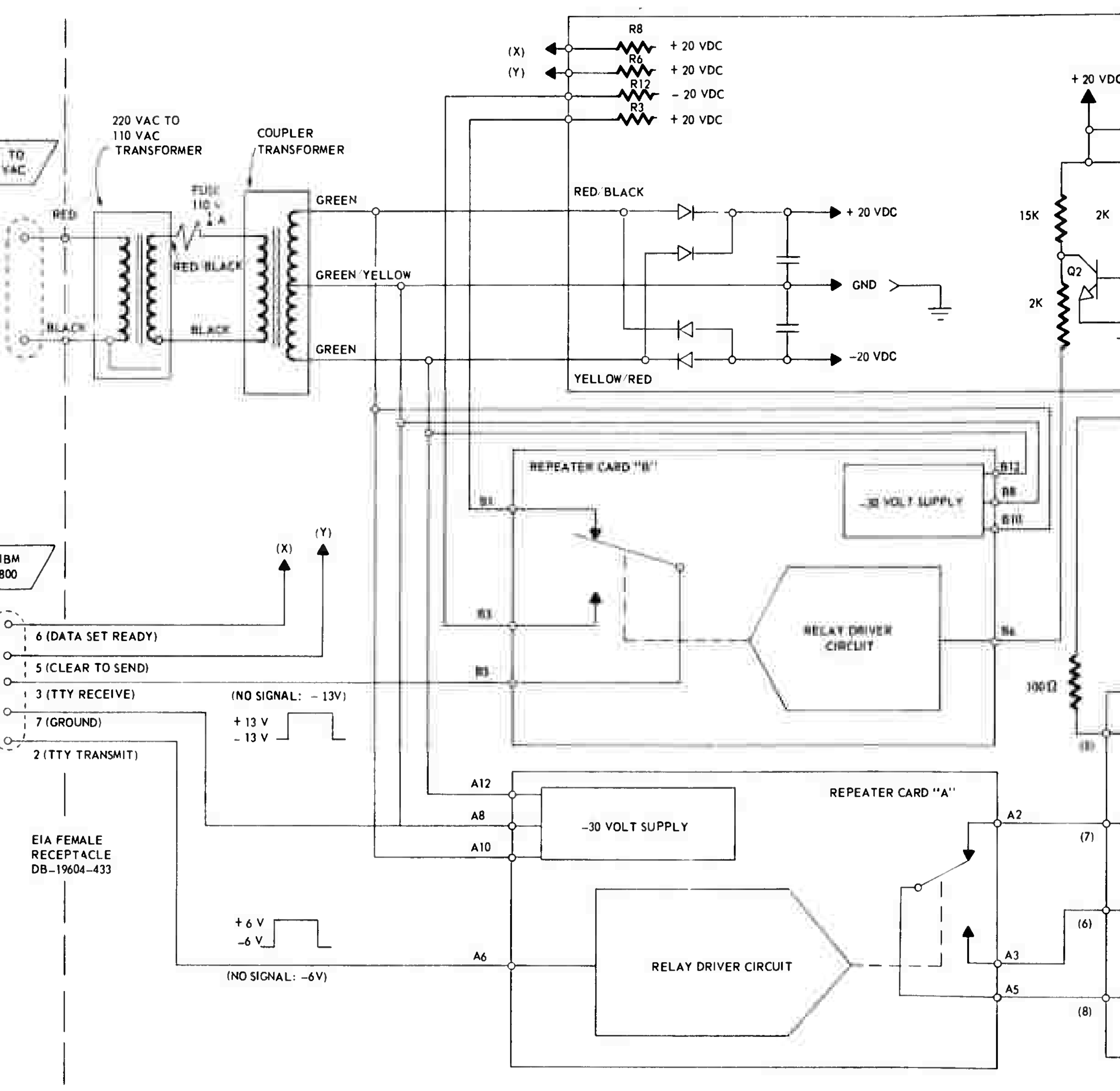
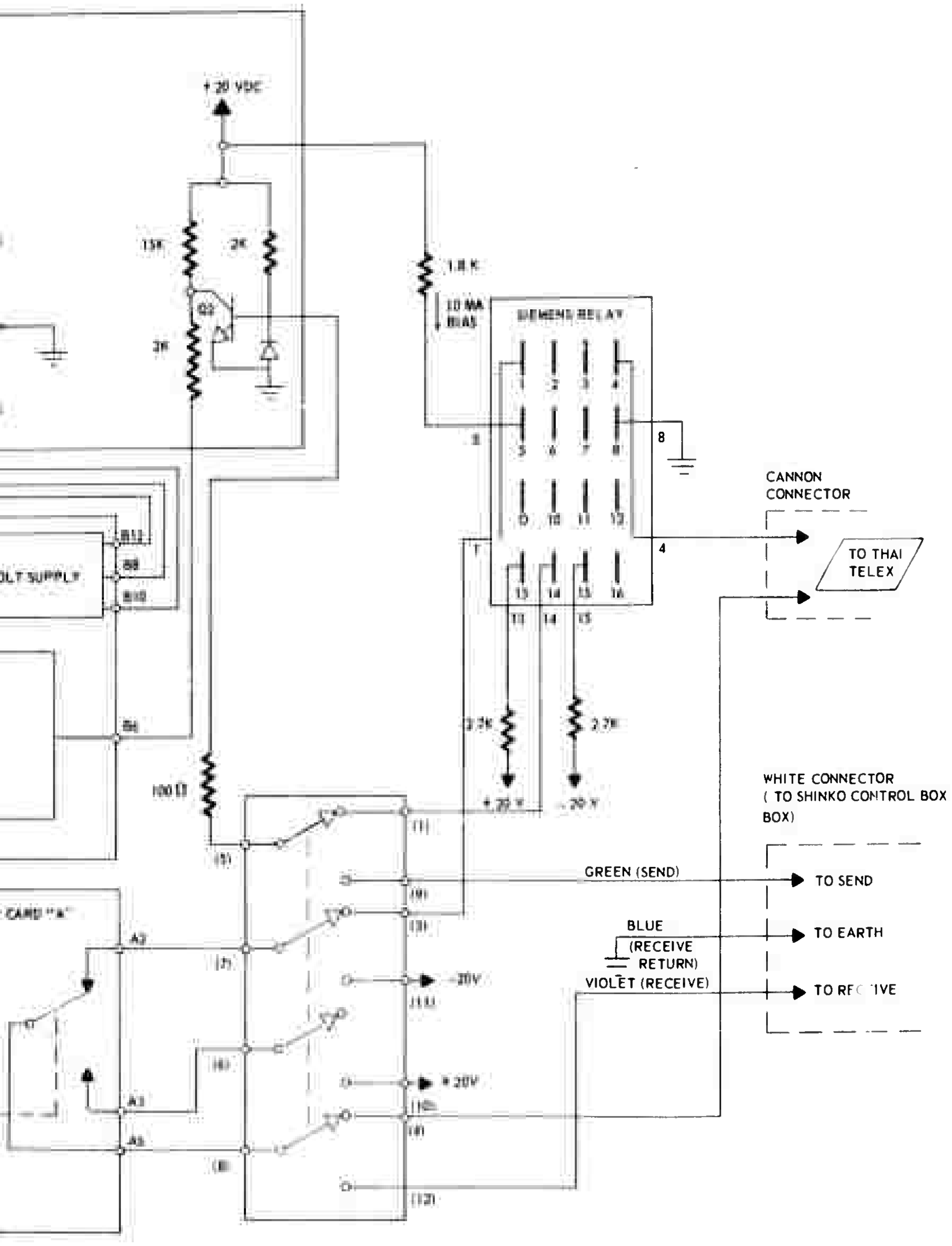


Figure D-2. Box #3 Schematic SDC Interface Box, 1800 to Thai Telex Interface or Shinko Control Box


+ 20 VDC
15K
2K
Q2
3K
1.8 K
10 MA BIAS
SIEMENS RELAY
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
8
4
CANNON CONNECTOR
TO THAI TELEX
B12
B8
B10
OLT SUPPLY
B6
2.7K
2.7K
+ 20 V
- 20 V
100 Ω
(1)
(5)
(9)
(3)
(7)
(11)
-20V
(6)
+20V
(10)
(4)
(8)
(12)
CARD "A"
A2
A3
A5
WHITE CONNECTOR (TO SHINKO CONTROL BOX BOX)
GREEN (SEND)
TO SEND
BLUE
(RECEIVE RETURN)
TO EARTH
VIOLET (RECEIVE)
TO REC IVE

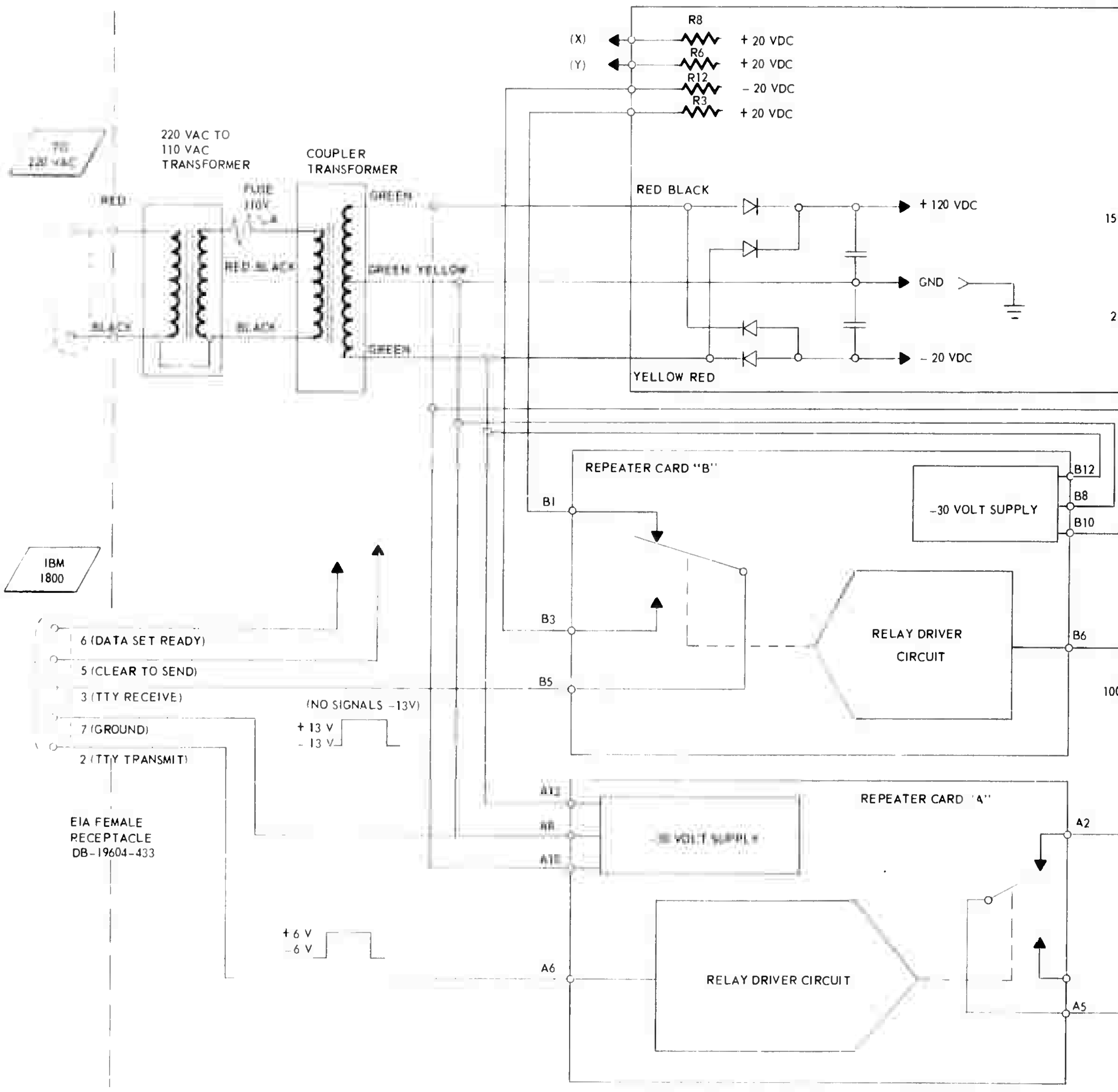
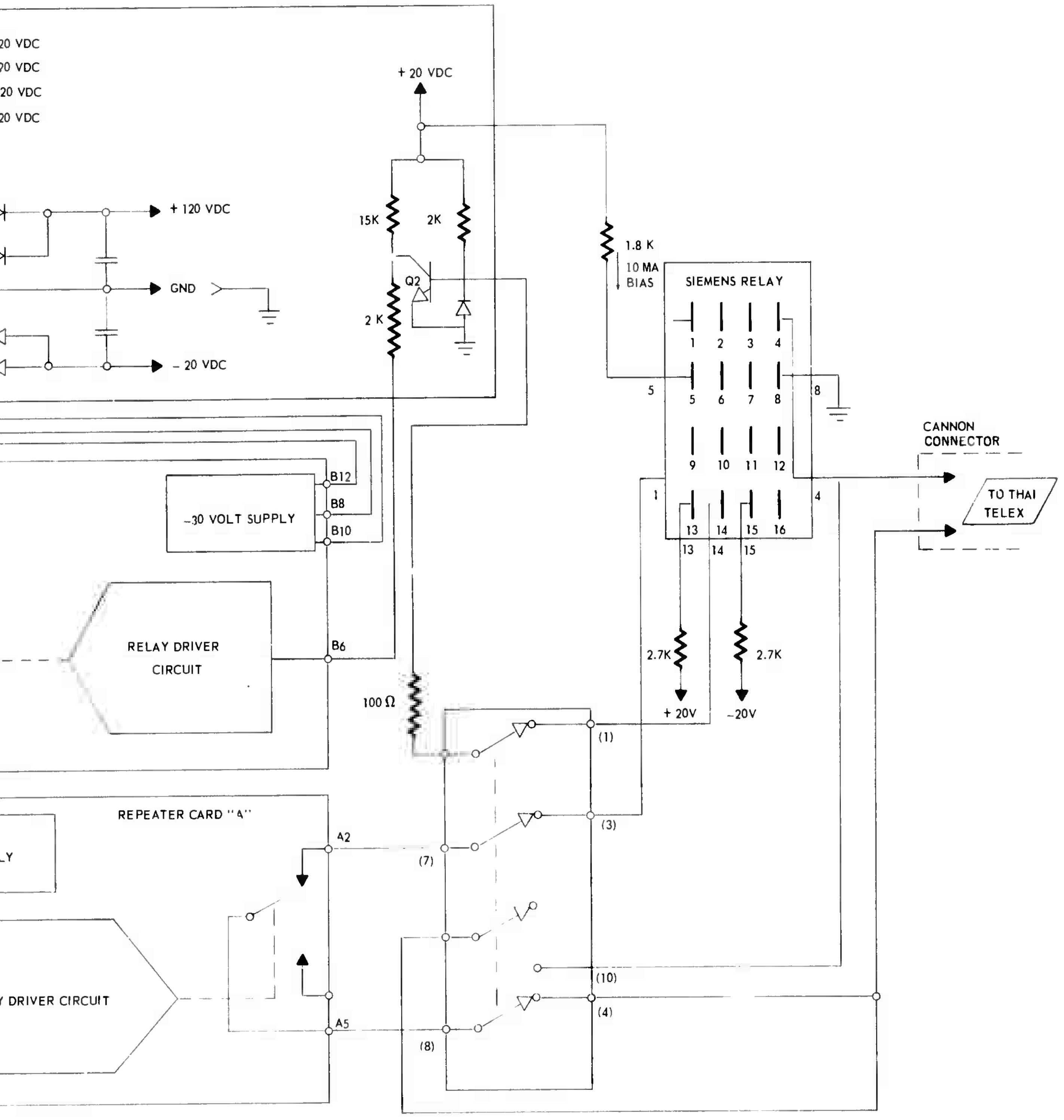


Figure D-3. Box #4 Schematic SDC Interface Box,
1800 to Thai Telex Interface


20 VDC
20 VDC
20 VDC
20 VDC
+ 20 VDC
+ 120 VDC
GND
– 20 VDC
15K
2K
2 K
Q2
1.8 K
10 MA BIAS
SIEMENS RELAY
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
CANNON CONNECTOR
TO THAI TELEX
B12
B8
B10
–30 VOLT SUPPLY
RELAY DRIVER CIRCUIT
B6
100 Ω
2.7K
2.7K
+ 20V
–20V
(1)
(3)
(7)
(10)
(4)
(8)
REPEATER CARD "A"
A2
A5
LY
DRIVER CIRCUIT

APPENDIX E

CLASS OUTLINES

The following is an outline of the general subjects presented in 24 three-hour sessions. The IBM 1800 computer was used by the students to assemble, run, and to debug their programs.

1. Problems Amendable to Solution by Computer: Discussion
2. General Computer Configuration - Information Flow
3. Memory: Addressing (direct)
   - load
   - store
   - add
   - subtract
4. Number Systems
   - binary
   - hexadecimal
   - octal
   - etc.
5. Practical Applications of Number Systems
   - addressing
   - computer arithmetic
   - instruction representation
6. Decision Making
   - flow diagramming introduction
   - conditional branch instructions
7. Simple Table Structures, Iteration and Subscripts
8. Coding Iterative Processes
   - index registers

9. Data Representation
   - signed and unsigned integers
   - floating point items
   - hollerith status items
   - boolean items
   - strings
   - trees
   - networks
   - pushdown lists (symmetric and other)
   - arrays
   - fixed field
   - floating field
10. The Assembly Process
    - translation
    - address assignment
    - pseudo operation codes
11. Sorting
12. Subroutines
    - indirect addressing
    - linkage
13. Retrieval Techniques
    - hashing
    - direct look up
    - binary search
14. Decision Tables
15. Input/Output: Interrupts
    - card reader
    - line printer
    - tape
    - punch
    - disk
16. Time-Sharing Systems

17. Programming with Fixed Point and Floating Point Data
18. Procedure-Oriented Languages
    - algebraic
    - string processing
    - list processing (recursion)
19. Compilers, Interpreters
20. Problem-Oriented Languages/Data Management Systems
21. Pattern Recognition and Learning Programs

MRDC ADVANCED PROGRAMMING LECTURES

Session I - Timesharing

1) Introduction
2) Interrupts
3) Paging
4) Terminals
    a) 2740
    b) Shinko
    c) KSR33

Session II - Timesharing

1) System Commands
2) Debugging
3) Demonstration of 1800 Resource-Sharing System
    a) Fortran-Editor
    b) Compiler
    c) Non-Technical Demonstration

Session III - Timesharing

1) Hands-on Experience with 1800
    a) Writing a Program
    b) Compiling
    c) Debugging

Session IV - Multiprogramming, Multiprocessing

1) Definitions
2) Data-Channels and Satellite Computers
3) Data Channel Control and Synchronization

Session V - Multiprogramming, Multiprocessing (continued)

4) Buffered Input/Output
5) Modularity of Processors
   a) Foreground and Background Machines
   b) Grosch's Law
6) Efficiency Versus Flexibility
7) Virtual Processors

Session VI - Multiprogramming, Multiprocessing (continued)

8) Virtual Computers
9) Measures of System Efficiency
10) Foreground and Background Processes
11) Paging and segmentation

Session VII - Telecommunications

1) Need for Increased Capacity
   a) Information Explosion
   b) Data Transmission in Business
   c) Public Use of Computers
2) Machines that use Data Transmission
3) Types of Lines
   a) Telephone Channels
   b) Simplex, Duplex, Half Duplex
   c) Voice and Subvoice

d) Telex

e) Modes of Transmission

- Parallel
- Start Stop
- Synchronous

Session VIII - Telecommunications (continued)

4) DC Signaling

5) AC Transmission Media

6) Channel Capacity

a) Bands

b) Signaling on a Channel with Noise

7) Modulation

a) Amplitude

b) Frequency

c) Phase

Session IX - Telecommunications (continued)

8) Multiplexing

a) Frequency Division

b) Time Division

9) Modems and Data Sets

10) Acoustical Coupling

11) Examples of Real-Time Computer Systems

Session X - Assemblers, Symbol Tables and Macros

1) Overall Structure of Assemblers

2) Operation Codes, Address Codes, Pseudo Operations

3) Symbol-Table Construction

4) Symbol Table Look Up

a) Hashing Techniques

b) Sorting and Binary Search

Session XI - Assemblers, etc. (continued)

5) Tree Structured Symbol Tables

6) Stack Structured Symbol Tables

7) Table Look Up

8) Table Driven Programs and Subprograms

9) Assembly with Macro Operations

a) Macro Definition

b) Macro Call

c) Macro Expansion

Session XII - Procedure-Oriented Languages

1) Data Definition

2) Statements and Expressions

3) Functions and Subprograms

4) Input/Output

5) Compilers, Interpreters

Session XIII - Data Management Systems/Problem-Oriented Languages

1) Introduction

2) Query Commands

3) Qualification

Session XIV - Data Management Systems (continued)

4) Format Control

5) Data Base Definition

6) Data Base Load

Session XV - Data Management Systems (continued)

7) Data Base Maintenance

8) Card Systems

9) Report Generator

10) Comparison of Data Base Systems

Sessions XVI and XVII - Seminar