OPTIMAL REALIZATION OF BENGALI KEY-BOARD AND CHARACTER ENCODING FOR COMPUTER APPLICATIONS

BY

MD. MOZAMMEL HUQ AZAD KHAN

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE IN ENGINEERING (COMPUTER)

DEPARTMENT OF COMPUTER ENGINEERING
FACULTY OF ELECTRICAL AND ELECTRONIC ENGINEERING
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DHAKA, BANGLADESH

OCTOBER,1986

CERTIFICATE

This is to certify that this Thesis work was done by me and it has not been submitted elsewhere for the award of any degree or diploma.

Signature of the Candidate

(MD. MOZAMMEL HUQ AZAD KHAN)

Countersigned

SMRahman 20/10/86

(DR. SYED MAHBUBUR RAHMAN)
Supervisor

Accepted as satisfactory for partial fulfilment of the requirements for the degree of Master of Science in Engineering (Computer), Bangladesh University of Engineering & Technology, Dhaka.

EXAMINERS

SMRahman 20/10/86

1. (DR. SYED MAHBUBUR RAHMAN)
Associate Professor,
Department of Computer
Engineering, BUET.

Chairman and
Supervisor

Ahmed 20/10/86

2. (DR. SHAMSUDDIN AHMED)
Head,
Department of Computer Engineering
&
Professor and Dean,
Faculty of Electrical and
Electronic Engineering, BUET.

Member

Dulal C. Kar 20/10/86

3. (MR. DULAL CHANDRA KAR)
Assistant Professor,
Department of Computer
Engineering, BUET.

Member

20/10/86

4. (DR. ABDUL MATIN PATWARI)
Vice-Chancellor,
Bangladesh University of
Engineering and Technology, Dhaka.

Member
External

ABSTRACT

The present work deals with the realization of Bengali keyboard and character encoding for computer applications. The only available Bengali key-board used for mechanical typewritting-the 'Optima Munir' is unsuitable for computer applications from different point of views, which have been studied in detail in the present research. For developing new Bengali key-board suitable for computer applications, a 434 Bengali Character Set has been identified among which 302 are Compound Byanjana Varnas having complex graphics. A 131 Bengali Graphic Symbol Set (BGS) has been selected by which the 434 Bengali characters can be generated. Based on the frequency of occurrence of the Bengali Graphic Symbols, two key-board lay-outs have been deviced, one with 56 main keys and other with 47 main keys, such that the load is equally distributed on all the active fingers. Both the key-board lay-outs have been deviced with two shift keys. A 95 Bengali Impression Symbol Set (BIS) have been selected by which 127 Bengali Graphic Symbols can be represented in typically used solid font printers if superposition is allowed. The 131 Bengali Graphic Symbols along with SPace, DELete and 32 ASCII standard control code have been encoded in an 8-bit coding scheme and a code mapping scheme has been proposed for the data communication systems which handle 7-bit characters. An algorithm has been developed for changing the notionally different English key-board, available with all microcomputers, to Bengali key-board and an assembly language interface program based on this algorithms has been developed for IBM PC Microcomputer.

ACKNOWLEDGEMENT

It is a matter of great pleasure on the part of the author to acknowledge his profound gratitude to his supervisor, Dr. Syed Mahbubur Rahman, Associate Professor, Department of Computer Engineering, BUET, for his advice, valuable guidence and constant encouragement throughout the progress of this work.

He wishes to express his thanks and deep sense of gratitude to Mr. Forhad Khan, Deputy Director, Bangla Academy, Dhaka for his kind help and for providing information related to this work.

The author is also indebted to Late Dr. A.K.M. Mahfuzur Rahman Khan, the founder head of the department of Computer Engineering for his inspiration throughout the work, who just left us few days before the completion of this work.

Thanks are due to Mr. Forhad Hossain, Scientific Officer, Institute of Computer Science, Bangladesh Atomic Energy Commission, Mr. Mallik Shameem Ahsan, Programmer,BUET Computer Center, Mr. Sanjoy Kumar Poddar, Programmer, BUET Computer Center, Mr. N.M. Jahangir, Analyst Programmer, ICDDR,Bangladesh, Mr. Dulal Chandra Kar, Assistant Professor,department of Computer Engineering, BUET and Mr. A.N.M. Mesbahul Karim, Asstt. Engineer, Institute of Computer Science, Bangladesh Atomic Energy Commission for their various helps during this work. He also wishes to give thanks to all others who directly or indirectly help the author during this work.

The author also expresses his sincere thanks to Mr. Hasan Ali and Mr. Yousuf Harun for taking patience care while typing and drafting this thesis.

CONTENTS

Page

CHAPTER 1

INTRODUCTION

1.1 GENERAL

About one thousand years ago, in this land of Bengal, the old Indian Arza- language had evolved a Prakrita- language - 'Bangla' or Bengali, which by the passage of this one thousand years has become the 6th language of the world on the basis of population. Bengali is,now, being spoken by more than 200 million peoples of Bangladesh, West Bengal and some parts of Asham. Official correspondence, bussiness transactions and education upto the level of University education, in this region, are being successfully performed by Bengali. The unimagining achievement of computer technology has so established its social impact that the use of computers is being rooted in the every-day life. To avail the blessing of this technological achievement, the use of Bengali as a medium of information interchange with a computer is needed.

1.2 FIELDS OF IMPLIMENTATION OF BENGALI IN COMPUTER APPLICATIONS

Bengali can be used as a medium of information interchange with an existing computer in graphics mode. But the notable disadvantages of graphics mode application are:

- it consumes more time for processing.
- it requires more internal memory space for information storage.

- it can not be used in high-speed applications, viz., real-time applications.
- it increases computer overhead.

To overcome these disadvantages, hard-wire implementation is required and the distinct fields of such implementation, though closely related and dependent upon each other, are:

- Text entry key-board
- Soft-copy display devices
- Hard-copy printing devices
- Internal representation and processing
- Data communication.

Proper implementation of Bengali in computer applications deserves equal attention and extensive research in all these fields.

1.3 OBJECTIVE OF THE PRESENT WORK

The important and primary thing, for implementation of Bengali in computer applications, is to design a standard text entry key-board and to encode the characters accomodated on the key-board such that the same characters and their codes can be used in soft-copy display devices, hand-copy printing devices, internal representation and processing and data communication. The objective of the present work is to design such a Bengali key-board and to encode the characters accomodated on the key-board.

1.4 WHY IS STANDARDIZATION OF BENGALI KEY-BOARD NEEDED ?

The ultimate goal of using a language in computer is to have a visual feedback of the text entered through the key-board on a video-display unit and to have a soft-copy display or a hard-copy printing of the processed text on a video-display unit or a hard-copy printer. Two other important things,though not visible to the high-level users,are interval representation and processing and data communication. Information flow in the pipe-line of text processing is shown in Fig. 1.1.

Design of a video-display unit and a hard-copy printer requires a set of predifined characters to be displayed or printed on the devices and their corresponding codes to be firm-wired in the devices. Internal representation and processing requires a set of internally representable codes of characters such that each character can be uniquely identified and processed. Data communication also requires a set of codes of characters such that during data transmission each character can be uniquely identified and transmission error can be detected for recognizing the original character transmitted.

Implementation of Bengali in all of the above fields requires a set of predifined characters and their corresponding codes by which Bengali language can be represented and processed by the above devices and processes. As key-board is the first

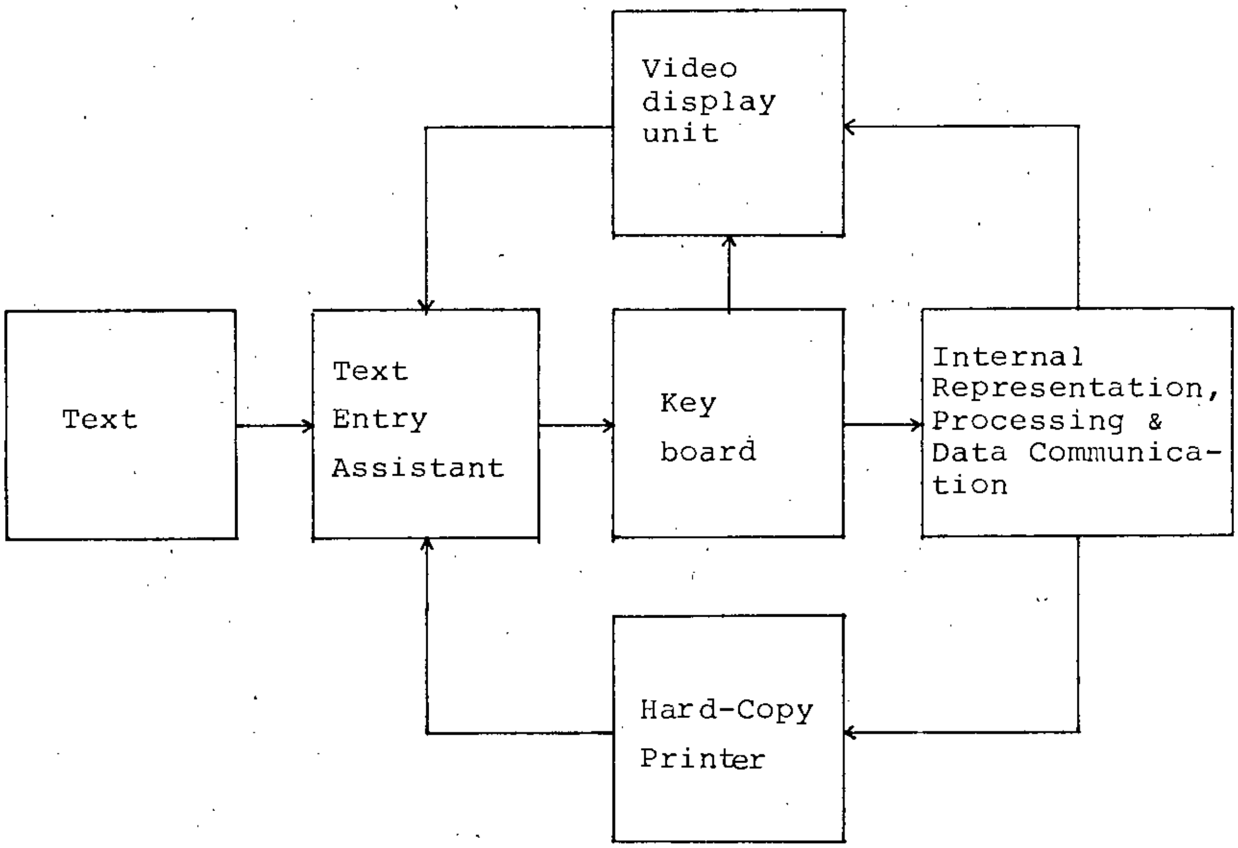


Fig. 11 : Information Flow in the Pipe-line of Text Processing.

step in the pipe-line of text processing, it should be provided with the predefined set of characters such that corresponding codes can be generated from the key-board for use in other devices and processes. Before giving attention to all other fields of implementation, the key-board should be made standard such that a coding scheme can be adopted for the characters accomodated on the key-board and independent persons can work on the other fields of implementation.

1.5 METHODOLOGY OF THE PRESENT WORK

To design a Bengali key-board, the following methodological procedures are required:

- Determination of a linguistically acceptable theoretical framework for selecting the key-board primitives, i.e., the elements corresponding to each key of a key-board by which a text is to be entered into a computer.
- Determination of total character set of Bengali language and selecting a graphic primiset such that:
  i) the primitives can be accomodated on a handy size of key-board,
  ii) the primitives can be used to represent Bengali text in the linguistically acceptable form, and
  iii) the primitives can be encoded by a standard size of code bits.

- Determination of key-board lay-out of the selected graphic primitives on the basis of the statistical analysis of frequency of occurrence of the primitives.
- Encoding the selected graphic primitives for internal representation and data communication and realization of codes from the designed key-board.

1.6 LITERATURE SURVEY OF RELATED WORKS

In 1976, Prabir Kumar Das [1] made a survey on the frequency of occurrence of the Bengali characters on the basis of 43126 no. of occurrence for computing the information content of the Bengali characters. But in this survey, Space has not been counted as a character though the Space is considered as a character in both communication and computer applications.

In 1984, Tapan Kumar Ghoshal et al.[2] developed a Bengali-Ahamia text processing attachment and named 'VIDYASAGAR'. This text processing system comprises of a text entry terminal with VDU, a dot-matrix printer and software routines. This system uses a Z-80-based Orion 8000 microcomputer with CP/M 2.2 operating system and the hardware attachments are connected to the host through industry standard RS-232C serial links to form a word processor, file creation unit or a file enquiry unit for Bengali/ Ahamia text. The coding scheme (BIIC) and the QWERTY style key-board employ simple consonants, vowels, operators and are accommodated in 7-bit, 96 character ASCII space. Host resident software are used to edit BIIC files, convert them to justified, paragraphed and paginated printable files with composite consonants, display the file in VDU or print them in the dot-matrix printer either with complex conjunct consonants or in analysed mode (without consonant conjuncts). In this text processing system, Vowel operators or Swara-kars are entered immediately after the consonant sequence on which they operate. But in Bengali text, some of the Vowel operators are placed at the left of the consonant sequence on which they operate and some are placed at the both side of the consonant sequence. Entering these Vowel operators after the consonant sequence will, obviously, be cumbersome for an ordinary text entry assistant. Compound consonants are entered by using conjuctive operator and the constituent consonant letters. But some of the compound consonant letters of Bengali language are found with unusual shapes, i.e., the constituent letters are not readily recognizable from their shapes. Entering these compound consonant letters by using conjunctive operator and constituent letters will be difficult for an ordinary text entry assistant.

In 1984, Sagar Mitra et al.[3] developed a character-mode raster-scan VDU for representing variable width composite consonant text. In this system, each symbol is split up into unit width components (sub symbols) and an n-unit width symbol is generated by n-sub symbols and the variable width phenomenon is tackled using a fixed width display. Motorola MC 6845 CRTC and (notionally incompatible) Intel 8085A MPU were used for designing the VDU.

In 1984, Gourhari Das et al.[4] proposed a 95 impression symbol set (BMIS) by which the 159 extended Sen and Datta graphic symbol set (SDBM) can be represented in Line printers and Daisy Wheel printers if superposition is allowed. As the SDBM does not contain all possible compound consonant letters with unusual shape, the BMIS is insufficient for generating all possible compound consonant letters. They also made a survey on the frequency of occurrence of the SDBM on the basis of 66,752 no. of occurrence.

Last year, the Bangla Academy had taken a project to improve the existing 'Optima Munir' Bengali key-board and collected opinions from general peoples. The opinions are now under consideration of an expert committee and no official decision has yet been published.

Researches have also been carried on in the department of Computer Engineering of the Bangladesh University of Engineering and Technology, Dhaka. Recently, Syed Mahbubur Rahman et al.[5,6] have developed methods for Bengali Alphanumeric dot matrix display. They used 14 x 8 dot matrix for representing unit width characters and 14 x 4 dot matrix for representing half width characters. Unit width character matrix is actually divided into two 14 x 4 half width matrices. For representing compound consonant letters, the unit is further divided into 4 horizontal sub-units. Representation of compound consonant letters are effected by superposition of units. A.N.M. Mesbahul Karim et al.[7] have designed a variable width character generator for displaying variable width texts in VDU. The system is experimented with the Bengali text and the variable width requirement of the Bengali characters are satisfied by splitting up the characters into fixed width sub-symbols.

CHAPTER 2

THEORETICAL FRAMEWORK DEVELOPMENT

2.1 INTRODUCTION

Key-board is the first step in the pipe-line of text processing. For selecting the key-board primitives, i.e., the elements corresponding to each key of a key-board by which a text is to be entered into a computer, a linguistically acceptable theoretical framework is needed. Total Bengali character set has been identified and such a framework is developed and discussed in this chapter.

2.2 BENGALI CHARACTER SET

Bengali character set consists of several types of characters, viz., Varnas or letters (Alphabet), Diacritical marks, Swara-kars or Vowel-operators, Compound byanjana varnas or compound consonant letters, Aksharas with unusual shape of swara-kars, Numerals, Punctuation marks and Special graphic symbols which are discussed in the following articles.

2.2.1 Bengali Alphabet

Bengali alphabet has 11 Swara-varnas or Vowel letters as listed in Table 2.1a. Beside these, one more swara varna , অ্যা or এ্যা , though not included in the alphabet, is used in Bengali script.

Bengali alphabet has 39 Byanjana Varnas or Consonant letters as listed in Table 2.1b.

2.2.2 Diacritical Marks

In Bengali script, 3 diacritical marks are used as listed in Table 2.1c.

Urdha comma (') is used to indicate umlanted ও ,i.e., ও' ,viz., ক'রে.

All Byanjana Varnas except ৎ , ং , ঃ are symbols of Byanjanas with inherent অ swara. To indicate only the Byanjana, Hashanta (্) is used with the Byanjana Varna, viz., ক্ .

Chandrabindu (ঁ) is used with a Swara to indicate it's nasal sound , viz., আঁ .

2.2.3 Swara-Kars or Vowel-operators

10 Swaras except অ , when operate on any consonant, change their original shapes and take their kar forms as listed in Table 2.1d.

2.2.4 Compound Byanjana Varnas or Compound Consonant Letters

One or more Byanjanas combine with another Byanjana with inherent অ swara leading to a Compound Byanjana Varna. Theoretically the numbers of Compound Byanjana Varnas are unlimited though a bulk majority of these may not be used in Bengali. 302 Compound Byanjana Varnas have been identified in use and in other cases byanjanas are conjuncted using hasanta (্). In practice Compound Byanjana Varans with two byanjanas, three byanjanas and four byanjanas are found. 222 Byanjana Varnas with two byanjanas, 74 Byanjana Varnas with three byanjanas and 6 Byanjana Varnas with four byanjanas have been identified and listed in Table 2.1e, 2.1f and 2.1g respectively.

2.2.5 Aksharas With Unusual Shape of Swara-Kars

25 Aksharas are found in use with unusual shape of Swara-kars as listed in Table 2.1h.

2.2.6 Numerals

10 numerals are used as listed in Table 2.1i.

2.2.7 Punctuation Marks

12 punctuation marks are used in Bengali script as listed in Table 2.1j.

2.2.8 Special Graphic Symbols

For business and scientific purposes 21 graphic symbols including arithmetic operators are used as listed in Table 2.1k.

TABLE 2.1a SWARA VARNAS

অ	আ	ই	ঈ	উ	ঊ	ঋ	এ	ঐ	ও	ঔ
1	2	3	4	5	6	7	8	9	10	11
অ্যা / এ্যা										
12										

TABLE 2.1b BYANJANA VARNAS

ক	খ	গ	ঘ	ঙ
13	14	15	16	17
চ	ছ	জ	ঝ	ঞ
18	19	20	21	22
ট	ঠ	ড	ঢ	ণ
23	24	25	26	27
ত	থ	দ	ধ	ন
28	29	30	31	32
প	ফ	ব	ভ	ম
33	34	35	36	37
য	র	ল	ব	শ
38	39	40	41	42
ষ	স	হ	য়	ড়
43	44	45	46	47
ঢ়	ৎ	ং	ঃ	
48	49	50	51	

TABLE 2.1c DIACRITICAL MARKS

র্	্	ঁ
52	53	54

TABLE 2.1d SWARA-KARS

া	ি	ী	ু	ূ	ৃ	ে	ৈ	ো	ৌ
55	56	57	58	59	60	61	62	63	64

TABLE 2.1e COMPOUND BYANJANA VARNAS WITH TWO BYANJANAS

2nd / 1st	ক	খ	গ	ঘ	ঙ	চ	ছ	জ	ঝ	ঞ	ট	ঠ	ড	ঢ	ণ	ত	থ	দ	ধ	ন	প	ফ	ব	ভ	ম	য	র	ল	শ	ষ	স	হ	য়	ড়	ঢ়	ৎ
ক	ক্ক 65	ক্খ 56									ক্ট 67					ক্ত 68							ক্ব 69		ক্ম 70	ক্য 71	ক্র 72	ক্ল 73		ক্ষ 74	ক্স 75					
খ																										খ্য 76	খ্র 77									
গ			গ্গ 78	গ্ঘ 79											গ্ণ 80				গ্ধ 81	গ্ন 82			গ্ব 83		গ্ম 84	গ্য 85	গ্র 86	গ্ল 87								
ঘ															ঘ্ণ 88					ঘ্ন 89						ঘ্য 90	ঘ্র 91									
ঙ	ঙ্ক 92	ঙ্খ 93	ঙ্গ 94	ঙ্ঘ 95																					ঙ্ম 96											
চ						চ্চ 97	চ্ছ 98			চ্ঞ 99																চ্য 100										
ছ																										ছ্য 101										
জ								জ্জ 102	জ্ঝ 103	জ্ঞ 104													জ্ব 105		জ্ম 106	জ্য 107	জ্র 108									
ঝ																																				
ঞ						ঞ্চ 109	ঞ্ছ 110	ঞ্জ 111	ঞ্ঝ 112																											
ট											ট্ট 113	ট্ঠ 114											ট্ব 115		ট্ম 116	ট্য 117	ট্র 118									
ঠ																										ঠ্য 119										
ড			ড্গ 120										ড্ড 121	ড্ঢ 122									ড্ব 123		ড্ম 124	ড্য 125	ড্র 126									
ঢ																										ঢ্য 127										
ণ											ণ্ট 128	ণ্ঠ 129	ণ্ড 130	ণ্ঢ 131	ণ্ণ 132					ণ্ন 133			ণ্ব 134		ণ্ম 135	ণ্য 136										
ত																ত্ত 137	ত্থ 138			ত্ন 139			ত্ব 140		ত্ম 141	ত্য 142	ত্র 143									
থ																							থ্ব 144			থ্য 145	থ্র 146									
দ			দ্গ 147	দ্ঘ 148														দ্দ 149	দ্ধ 150				দ্ব 151	দ্ভ 152	দ্ম 153	দ্য 154	দ্র 155									
ধ																				ধ্ন 156			ধ্ব 157		ধ্ম 158	ধ্য 159	ধ্র 160									
ন											ন্ট 161	ন্ঠ 162	ন্ড 163			ন্ত 164	ন্থ 165	ন্দ 166	ন্ধ 167	ন্ন 168			ন্ব 169		ন্ম 170	ন্য 171					ন্স 172					
প											প্ট 173					প্ত 174				প্ন 175	প্প 176	প্ফ 177			প্ম 178	প্য 179	প্র 180	প্ল 181			প্স 182					
ফ																										ফ্য 183	ফ্র 184	ফ্ল 185								
ব								ব্জ 186										ব্দ 187	ব্ধ 188				ব্ব 189	ব্ভ 190		ব্য 191	ব্র 192	ব্ল 193								
ভ																										ভ্য 194	ভ্র 195	ভ্ল 196								
ম																				ম্ন 197	ম্প 198	ম্ফ 199	ম্ব 200	ম্ভ 201	ম্ম 202	ম্য 203	ম্র 204	ম্ল 205								
য																							য্ব 206			য্য 207										
র	র্ক 208	র্খ 209	র্গ 210	র্ঘ 211		র্চ 212	র্ছ 213	র্জ 214	র্ঝ 215		র্ট 216		র্ড 217		র্ণ 218	র্ত 219	র্থ 220	র্দ 221	র্ধ 222	র্ন 223	র্প 224	র্ফ 225	র্ব 226	র্ভ 227	র্ম 228	র্য 229		র্ল 230	র্শ 231	র্ষ 232	র্স 233	র্হ 234				র্ৎ 235
ল	ল্ক 236		ল্গ 237								ল্ট 238		ল্ড 239								ল্প 240	ল্ফ 241	ল্ব 242	ল্ভ 243	ল্ম 244	ল্য 245		ল্ল 246				ল্হ 247				
শ						শ্চ 248	শ্ছ 249													শ্ন 250			শ্ব 251		শ্ম 252	শ্য 253	শ্র 254	শ্ল 255	শ্শ 256							
ষ	ষ্ক 257									ষ্ঞ 258	ষ্ট 259	ষ্ঠ 260			ষ্ণ 261						ষ্প 262	ষ্ফ 263	ষ্ব 264		ষ্ম 265	ষ্য 266										
স	স্ক 267	স্খ 268									স্ট 269					স্ত 270	স্থ 271			স্ন 272	স্প 273	স্ফ 274	স্ব 275		স্ম 276	স্য 277	স্র 278	স্ল 279								
হ															হ্ণ 280					হ্ন 281			হ্ব 282		হ্ম 283	হ্য 284	হ্র 285	হ্ল 286								
য়																																				
ড়																																				
ঢ়																																				

TABLE 2.1f COMPOUND BYANJANA VARNAS WITH THREE BYANJANAS

ঙ্ক্ষ											
287											
ক্ষ্ণ											
288											
ক্ষ্ম											
289											
ক্র্য	ক্ল্য	ক্ষ্য	গ্ন্য	গ্র্য	ঙ্ক্য	ঙ্খ্য	ঙ্ঘ্য	চ্ছ্য	ণ্ড্য	ত্ম্য	
290	291	292	293	294	295	296	297	298	299	300	
ত্র্য	দ্ব্য	ন্ড্য	ন্দ্য	ন্ধ্য	ন্ন্য	প্ল্য	ষ্ক্য	ষ্ট্য	ষ্ঠ্য	ষ্ম্য	
301	302	303	304	305	306	307	308	309	310	311	
স্ট্য	স্ড্য										
312	313										
ত্ত্র	ক্ত্র	ঙ্ক্র	ঙ্গ্র	চ্ছ্র	ণ্ড্র	ত্ম্র	ন্ত্র	ন্দ্র	ন্ধ্র	ম্প্র	
314	315	316	317	318	319	320	321	322	323	324	
ম্ভ্র	ষ্ক্র	ষ্ট্র	ষ্প্র	স্ট্র	স্ত্র	স্প্র					
325	326	327	328	329	330	331					
ক্ষ্ব	চ্ছ্ব	জ্জ্ব	ত্ত্ব	ন্ত্ব	ন্দ্ব						
332	333	334	335	336	337						
র্ঘ্য	র্চ্চ	র্চ্ছ	র্জ্জ	র্ঢ্য	র্ণ্য	র্ত্ত	র্ত্য	র্থ্য	র্দ্দ	র্দ্ধ	র্দ্ব
338	339	340	341	342	343	344	345	346	347	348	349
র্দ্য	র্দ্র	র্ধ্ব	র্ব্ব	র্ব্ভ	র্ব্য	র্ষ্ম	র্য্য	র্শ্ব	র্ষ্ক	র্ষ্য	
350	351	352	353	354	355	356	357	358	359	360	

TABLE 2.1g COMPOUND BYANJANA VARNAS WITH FOUR BYANJANAS

র্দ্দ্য	র্ষ্ম্য	
361	362	
ক্ষ্ম্য	ন্ত্র্য	ঙ্ক্ষ্য
363	364	365
র্দ্ধ্ব		
366		

TABLE 2.1h AKSHARAS WITH UNUSUAL SHAPE OF SWARA-KARS

Swara kar / Varnas	ু	ূ	ৃ
গ	গু 367		
র	রু 368	রূ 369	
শ	শু 370		
হ	হু 371		হৃ 372
ন্ত	ন্তু 373		
ল্গ	ল্গু 374		
স্ত	স্তু 375		
ত্র	ত্রু 376	ত্রূ 377	
দ্র	দ্রু 378	দ্রূ 379	
ধ্র	ধ্রু 380	ধ্রূ 381	
প্র	প্রু 382	প্রূ 383	
ব্র	ব্রু 384	ব্রূ 385	
ভ্র	ভ্রু 386	ভ্রূ 387	
শ্র	শ্রু 388	শ্রূ 389	
স্র	স্রু 390	স্রূ 391	

TABLE 2.1i NUMERALS

০	১	২	৩	৪	৫	৬	৭	৮	৯
392	393	394	395	396	397	398	399	400	401

TABLE 2.1j PUNCTUATION MARKS

,	;	:	।	?	!	–	—
402	403	404	405	406	407	408	409
“	”	‘	’				
410	411	412	413				

TABLE 2.1k SPECIAL GRAPHIC SYMBOLS

#	৳	+	−	×	<	>	/	%	*	.
414	415	416	417	418	419	420	421	422	423	424
=	&	@	(	)	{	}	[	]	_	
425	426	427	428	429	430	431	432	433	434	

Though # , & and @ symbols are not normally used in Bengali texts, they have been included in this list for business and scientific purposes, specially for computer applications as delemeters.

2.3 LEXICAL PRIMITIVES

Lexical primitives are the elements with which a word is lexically analysed and ordered. Conversely, a sequence of lexical primitives corresponds to a unique word. Varnas of the Bengali alphabet correspond to the Lexical primitives of Bengali language. The set of Bengali Lexical primitives is divided into two subsets- Swaras or vowels and Byanjanas or consonants. The Swara subset consists of 11 Swaras and the Byanjana subset consists of 39 Byanjanas, which combinedly form the Bengali Lexical primiset.

2.4 AKSHARAS

Traditional visual representation of a Bengali word is formed by a sequence of 'Aksharas'. The aksharas are generated by special sequences of lexical primitives called Syllables.

There are four classes of Bengali syllables:

- Mono-vowel Syllables which correspond to all Swaras.
- Single consonant-vowel Syllables which correspond to all Byanjanas, except ৎ , ং , ঃ, with inherent অ-swara or operated on by other swara-kars or vowel operators.
- Single consonant Syllables which correspond to all Byanjanas.

- Poly-consonant-vowel Syllables which correspond to all Compound Byanjanas with inherent অ -swara or operated on by other swara-kars.

The visual forms of lexical primitives are called Varnas and the visual forms of syllables are called Aksharas.

2.5 GRAPHIC MOLECULES

Visual form of all syllables, i.e., all aksharas, all numerals, punctuation marks and special graphic symbols combinedly form the set of Graphic Molecules.

The possible mono-vowel aksharas are generated by 11 swara varnas, অ্যা or এ্যা (though they are phonetically same, they are equally used in Bengali script), umlanted ঔ and their various nasal forms, i.e., operated on by the chandrabindu. The possible single consonant-vowel aksharas are generated by 36 byanjana varnas, except ৎ , ং , ঃ , with inherent অ -swara and operated on by other swara-kars and their nasal forms. The possible single consonant aksharas are generated by ৎ , ং , ঃ and other 36 byanjana varnas operated on by the hashanta. The number of possible poly-consonant- vowel syllables are theoretically infinite and for that the number of possible poly-consonant-vowel aksharas are, also, theoretically infinite. If those compound byanjanas which are conjuncted by hashanta are treated as combination of seperate single consonant aksharas and single consonant-vowel aksharas, then the possible poly-consonant-vowel aksharas are generated by the identified 302 compound byanjana varnas with inherent অ-swara and operated on by other swara-kars and their nasal forms. With numerals, punctuation marks and special graphic symbols, the theoretically possible number of Graphic Molecules might be inconveniently large, though a bulk majority of these may not be used in Bengali.

2.6 GRAPHIC PRIMITIVES

The number of theoretically possible graphic molecules are inconveniently large. Though a bulk majority of these may not be used in Bengali, the usable set of graphic molecules is still inconveniently large. The graphic molecules can, however, be generated from a smaller set of graphics, using graphic transformation. The simplest graphic transformation is concatenation, i.e., placing one symbol after another. Bengali language generate the large number of graphic molecules by concatenating a smaller number of graphic symbols. Such graphic symbols are called graphic primitives. Selection of the graphic primiset is discussed in Chapter 3.

2.7 IMPRESSION PRIMITIVES

The Bengali graphic primitives can, however, be generated by a smaller set of impression symbols if superposition is allowed. These impression symbols are called impression primitives. Selection of the impression primiset is discussed in Chapter 4.

2.8 KEY-BOARD PRIMITIVES

The elements corresponding to each key of a Key-board by which a text is entered into a computer is called Key-board primitives. For one-to-one correspondence between input and output, the Bengali graphic primitives are needed to be used as Key-board primitives for entering text into a computer. The Bengali lexical primitives can, also, be used as Key-board primitives, but in this case, a complex mapping will be needed for realizing the possible graphic molecules. Selection of the Key-board primiset and its Key-board lay-out are discussed in Chapter 3.

2.9 INFORMATION PRIMITIVES

The entities that are used for entering a text into a computer is called the Information primitives. An important aspect of Information primitives is that each primitive will have a unique numeric code needed for machine representation and a unique graphic needed for visual feedback at the text-entry terminal. Information primiset should also contain appropriate control codes for computer usage.

While a text is entered through a key-board into a computer, the keys correspond to the Bengali graphic symbols selected for entering text into a computer and therefore are Information primitives. The coding scheme of the Information primiset is discussed in Chapter 5.

CHAPTER 3

BENGALI KEY-BOARD PRIMITIVES SELECTION AND KEY-BOARD LAY-OUT

3.1 INTRODUCTION

Though a bulk majority of the theoretically possible graphic molecules may not be used in Bengali texts, the usable set of graphic molucules is still inconveniently large and their number is not even specific and predictable. The unpredictable set of graphic molecules is traditionally mapped in letter press printing by about 500 characters or graphic primitives. The number is reduced by simplification, e.g., by Linotype. In 1982, Sen & Datta [8] proposed a 142 graphic symbol set which can adequately represent the Bengali language. Gourhari Das et al.[4] had extended the Sen & Datta's graphic symbol set to 159 symbols (appendix-A) to include special symbols, punctuation marks,arithmetic and logical operators required for commercial and technical work.

In computer applications, printing of a Bengali text is not only the point of consideration. Inputing the text into the computer also plays a vital role in this work. That is why, a correspondence between the input and the output is required, i.e., a correspondence should be made between the key-board primitives and the graphic primitives for printing the text. As a visual feedback is needed at the text entry terminal during the text entry through the key-board, a one-to-one correspondence between the key-board primitives and the graphic primitives is desirable, i.e., the key-board primitives and the graphic primitives are desired to be identical.

Selection of graphic primitives, key-board primitives and their key-board lay-out is discussed in this chapter.

3.2 SELECTION CRITERIA OF KEY-BOARD PRIMITIVES

As the Bengali compound byanjana varnas are of various shapes, generation of the Bengali graphic molecules by them becomes complex. The Bengali graphic molecules can, however, be generated by two methods. In one method, the generation of the graphic molecules can be done under software control. In this method, the compound byanjana varnas and aksharas with unusual shape are to be kept themselves as key-board primitives and other compound byanjana varnas are to be generated from their constituent varnas under software control. Obviously, this method will be complex, time consuming and cumbersome. In the other method, the shape of the compound byanjana varnas and aksharas are needed to be slightly changed in linguistically and aesthetically acceptable form such that the graphic molecules can be generated directly by concataneting a smaller set of graphic primitives. In this method, the graphic primitives can be used as the key-board primitives and a one-to-one correspondence between input and output will be possible.

Key-board primitives selection for two methods are discussed in the following articles.

3.3 KEY-BOARD PRIMITIVES SELECTION FOR SOFTWARE MAPPED GRAPHIC MOLECULES

All possible Bengali graphic molecules can be mapped by the 434 set of Bengali characters identified in Table 2.1a to 2.1k under software control. But the [illegible] set of characters would not be accomodated on a handy size of key-board and these characters will need 2 bytes ($\frac{\text{ld } 434}{8} = 1.095$ i.e.,2)for encoding them for machine representation. The number of key-board primitives can, however, be reduced if some complex algorithm is adapted for mapping graphic molecules.

The Swara Varnas of Table 2.1a can be mapped by the key-board primitives of Table 3.1a. আ is to be mapped by অ and া of Table 3.1d. অ্যা or এ্যা are to be mapped by অ or এ, ্য of Table 3.1e and া of Table 3.1d. The algorithm of mapping these Swara Varnas is simple and given in Fig. 3.1a.

The Byanjana Varnas of Table 2.1b can be mapped by the key-board primitives of Table 3.1b. The shape of ব and internal ব are similar and for that, the internal ব can be mapped by the ব of Table 3.1b. Mapping of Byanjana Varnas is direct and simple and the algorithm is given in Fig. 3.1a.

The Diacritical Marks of Table 2.1c can be used for mapping graphic molecules by the key-board primitives of Table 3.1c.

To place ্ and ঁ at the bottom and at the top of a varna respectively, a left conjunction operator is needed. ’ is to be entered as normal and ্ and ঁ are to be entered with a prefix left operator. A software routine then place them at the appropriate position. The algorithm is a little bit complex and given in Fig. 3.1b.

The Swara-Kars of Table 2.1d can be used with byanjana varnas by the key-board primitives of Table 3.1d. ো is to be mapped by ে and া and ৌ is to be mapped by ে and ৗ. For placing ু, ূ and ৃ at the bottom of the concerned byanjana varnas, a left conjunction operator is needed. These kars are to be entered with this prefix left operator and others are to be entered as normal. A software routine then place them at the appropriate position. The algorithm is a little bit complex and given in Fig. 3.1c.

Mapping the compound Byanjana Varnas of Table 2.1e to 2.1g are much complex. A first approach can be made by using a conjunctive operator with the constituent byanjana varnas, i.e., for mapping the Compound Byanjana Varnas, the constituent byanjana varnas are to be entered in their normal shape with a conjunctive operator in between them. A software routine then decides the shape of the Compound Byanjana Varnas by searching a table of Compound Byanjana Varnas. In this method, a three level of conjunction will be needed for mapping Compound Byanjana Varnas with two byanjanas, three byanjanas and four byanjanas. A more complex algorithm will be needed in this method and such an algorithm is given in Fig. 3.1d. A notable problem will arise with this method that a considerable number of Compound Byanjana Varnas are of unusual shape, i.e., the constituent varnas of these Compound Byanjana Varnas are not readily recognizable from their shape. Entering these Compound Byanjana Varnas by their constituent byanjana varnas and a conjunctive operator will be problematic and slow for an ordinary text entry assistant. This problem can, however, be overcome by a second approach, where Compound Byanjana Varnas with unusual shape themselves are to be kept as key-board primitives and other Compound Byanjana Varnas are to be mapped by their constituent byanjana varnas and a conjunctive operator by an algorithm similar to that of Fig. 3.1d. As, in this method, the Compound Byanjana Varnas are to be mapped by searching a table under a complex algorithm, the process will be considerably slow reducing the text entry speed. The text entry speed can, however, be improved by reducing the algorithmic complexity and a third approach can be made by keeping Compound Byanjana Varnas with unusual shape themselves as key-board primitives and analyzing the conjunction of other Compound Byanjana Varnas for devicing a simpler algorithm. Conjunction analysis of the Compound Byanjana Varnas with two byanjanas, three byanjanas and four byanjanas of Table 2.1e, 2.1f and 2.1g are given in Table 3.2a, 3.2b and 3.2c respectively. This analysis reveals that 31 compound byanjana varnas are of unusual shape and ref (ʼ), ন-fala, ল-fala, ব-fala, ম-fala, য-fala, র-fala and ণ-fala can be conjuncted as separate entities. At the first place, byanjana varnas conjunct at three positions, viz., at the normal position, at the top position and at the left-top position. At the second place, byanjana varnas conjuct at three positions, viz., at the right position, at the bottom position and at the right-bottom position. In the third approach of mapping Compound Byanjana Varnas, 31 compound byanjana varnas with unusual shape, 1 ref and 7 falas can be kept as key-board primitives. For ref and 5 falas, except ন-fala and ্য-fala, a left operator can be introduced. When these symbols are entered with prefix left operator, a software routine then display them at the appropriate position. In other cases, five position operators, two for the first place and three for the second place, viz., top, left-top, right, bottom and right-bottom respectively can be introduced. For entering compound byanjana varnas, each constituent byanjana varnas of normal shape are to be entered as normal or with an appropriate prefix operator. A software routine then display the constituent byanjana varnas at the appropriate position of the compound byanjana varna . As no table searching is needed in this approach ,algorithmic complexity will be simpler than that of the second approach. Such an algorithm is given in Fig. 3.1e. But the algorithm is still complex and the text entry speed will be reduced by a huge number of position operators. Moreover, various shapes and positions of a single character are to be decided by this algorithm from the same key-board input. These problems can, however, be reduced by a fourth and final approach, where the number of the position operators is to be reduced by introducing additional symbols on the basis of the conjuction analysis of the Compound Byanjana Varnas of Table 3.2a to 3.2c. For conjunction at the normal position, at the right position and at the right-bottom position, key-board primitives of Table 3.1b are adequately sufficient. For conjuction at the top position and at the left-top position, 17 small sized symbols of byanjana varnas are required (Table 3.1e). For conjunction at the bottom position, 12 small sized symbols of byanjana varnas without'matra' are required. Among these, ম, ন , ল and ব are common with the symbols required for conjunction at the top position and at the left-top position. Other 8 such symbols are required (Table 3.1e). Ref is required as a separate entity. Among 7 falas, ন is common with the symbols required for conjuction at the top position and at the left-top position. ল, ব and ম are common with the symbols required for conjunction at the bottom position. Symbols for other 3 falas are required (Table 3.1e). 31 Compound Byanjana Varnas with unusual shape themselves are required to be kept as key-board primitives. All these key-board primitives for mapping Compound Byanjana Varnas are listed in Table 3.1e. In this final approach, three conjunction operators, viz., bottom operator,left operator, and right operator, will be required. Symbols for normal position, top position, left-top position and compound byanjana varnas with unusual shape are to be entered as normal. Symbols for ref and 5 falas other than ্য and র্ are to be entered with a preffix left operator. Symbols for right position, right-bottom position and্য&র্are to be entered with a preffix right operator. Symbols for bottom position are to be entered with a preffix bottom operator. A software routine then display the concerned Compound Byanjana Varnas. The algorithm of this final approach is given in Fig. 3.1f.

The Aksharas with unusual shape of swara-kars of Table 2.1h can be mapped by the key-board primitives of Table 3.1f. Among the 10 swara-kars, only ু-kar, ূ-kar and ৃ-kar change their shapes during operation on some specific byanjana varnas. A variant of ু-kar, i.e., ᵕ , can be introduced as a key-board primitive and other aksharas with unusual shape of ু-kar are required to be kept as key-board primitives. For ূ-kar, a variant, i.e., ℸ, can also be introduced as a key-board primitive. The ৃ-kar only changes its shape during operation on হ and this akshara is kept as a key-board primitive, because no benifit can be obtained by introducing a seperate variant of ৃ-kar. The variants of ু-kar and ূ-kar, i.e., ᵕ and ℸ , are to be entered seperately at the right of the concerned byanjana varnas. Mapping of these aksharas with unusual shape of swara-kars is direct and simple and the algorithm is given in Fig. 3.1a.

The Numerals, Punctuation Marks and Special Graphic Symbols of Table 2.1i, 2.1j and 2.1k can be mapped by the key-board primitives of Table 3.1g, 3.1h and 3.1i respectively. The mapping is direct and simple and the algorithm is given in Fig. 3.1a.

In this method of mapping graphic molecules under software control, a 172 key-board primiset of Table 3.1a to 3.1i is required, which is much to be accomodated on a handy size of key-board. If these 172 key-board primitives are accomodated on a key-board with 47 main keys, 4 symbols are required to be assigned to each key (one normal and three shift symbols) with three conjunction operators, viz., bottom, left and right operators. Such a key-board lay-out might be as given in Fig. 3.2.

This method of mapping graphic molecules under software control has the advantage that the graphic molecules can be mapped in the conventional form. But this method does have a lot of notable problems as follows:

- The numbers of key-board primitives are 172. For accomodating these 172 key-board primitives on a key-board with 47 main keys, 4 symbols are required to be assigned to each key, which is much problematic for human eye, because the human eye can identify at most three entities as a group at a time.

TABLE 3.1a KEY-BOARD PRIMITIVES FOR SWARA VARNAS FOR SOFTWARE MAPPED GRAPHIC MOLECULES

অ ই ঈ উ ঊ ঋ এ ঐ ও ঔ

TABLE 3.1b KEY-BOARD PRIMITIVES FOR BYANJANA VARNAS FOR SOFTWARE MAPPED GRAPHIC MOLECULES

ক	খ	গ	ঘ	ঙ
চ	ছ	জ	ঝ	ঞ
ট	ঠ	ড	ঢ	ণ
ত	থ	দ	ধ	ন
প	ফ	ব	ভ	ম
য	র	ল		শ
ষ	স	হ	য়	ড়
ঢ়	ৎ	ং	ঃ	

TABLE 3.1c KEY-BOARD PRIMITIVES FOR DIACRITICAL MARKS FOR SOFTWARE MAPPED GRAPHIC MOLECULES

TABLE 3.1d KEY-BOARD PRIMITIVES FOR SWARA-KARS FOR SOFTWARE MAPPED GRAPHIC MOLECULES

TABLE 3.1e KEY-BOARD PRIMITIVES FOR MAPPING COMPOUND BYANJANA VARNAS FOR SOFTWARE MAPPED GRAPHIC MOLECULES

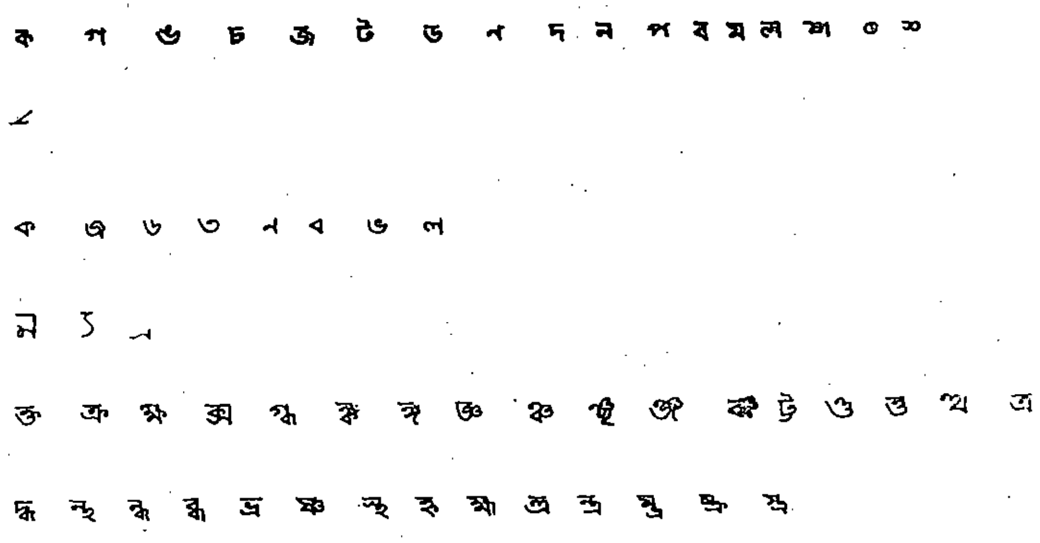


TABLE 3.1f KEY-BOARD PRIMITIVES FOR MAPPING AKSHARAS WITH UNUSUAL SHAPE OF SWARA-KARS FOR SOFTWARE MAPPED GRAPHIC MOLECULES

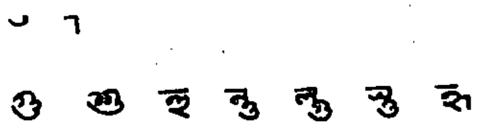


TABLE 3.1g KEY-BOARD PRIMITIVES FOR NUMERALS FOR SOFTWARE MAPPED GRAPHIC MOLECULES

০ ১ ২ ৩ ৪ ৫ ৬ ৭ ৮ ৯

TABLE 3.1h KEY-BOARD PRIMITIVES FOR PUNCTUATION MARKS FOR SOFTWARE MAPPED GRAPHIC MOLECULES

, ; : । ? ! – — “ ” ‘ ’

TABLE 3.1i KEY-BOARD PRIMITIVES FOR SPECIAL GRAPHIC SYMBOLS FOR SOFTWARE MAPPED GRAPHIC MOLECULES

৳ + − × < > / % * . = & @

( ) { } [ ] _

TABLE 3.2a CONJUNCTION ANALYSIS OF COMPOUND BYANJANA VARNAS WITH TWO BYANJANAS

Varnas	Conjunction at the first place					Conjunction at the second place					Remarks
	Normal Position	Top Position	Left-Top Position	Other special Position	Unusual Shape	Right Position	Bottom Position	Right-Bottom position	Other special position	Unusual Shape	
ক	ক্ব	ক্ক	ক্থ		ক্ত ক্র ক্ষ ক্ম		ক্ক			ঙ্ক	
খ	খ্র					ক্খ					
গ	গ্য	গ্ন	গ্ঘ		গ্ধ	ড্গ	গ্গ			ঙ্গ	
ঘ	ঘ্ন					গ্ঘ					
ঙ	ঙ্ম		ঙ্খ		ঙ্ক ঙ্ক্ষ						ঙ does not conjunct at the second position
চ	চ্য		চ্চ			চ্চ				ঞ্চ	
ছ	ছ্য					চ্ছ				ঞ্ছ	
জ	জ্ব		জ্জ		জ্ঞ		জ্জ	জ্জ		ঞ্জ	
ঝ						জ্ঝ				ঞ্ঝ	ঝ does not conjunct at the first position
ঞ					ঞ্চ ঞ্ছ ঞ্জ ঞ্ঝ	চ্ঞ				জ্ঞ	with ঞ at the first position all are of unusual shape
ট	ট্ব		ট্ট		ট্র	ক্ট				ষ্ট	
ঠ	ঠ্য					ট্ঠ					
ড	ড্ব		ড্গ			ড্ড	ন্ড			ণ্ড	
ঢ	ঢ্য					ড্ঢ					
ণ	ণ্য	ণ্ন	ণ্ট		ণ্ড		গ্ণ		ম্ণ	ষ্ণ	ণ fala
ত	ত্ব				ত্ত ত্থ ত্র		ন্ত			ক্ত ত্ত	
থ	থ্ব									ত্থ ন্থ স্থ	with থ at second position all except ত্থ are of unusual shape
দ	দ্ম	দ্ব	দ্গ		দ্ধ	ন্দ		দ্দ			
ধ	ধ্ন									গ্ধ দ্ধ ব্ধ ন্ধ	with ধ at the second position all except দ্ধ are of unusual shape

TABLE 3.2a CONTD.

Varnas	Conjunction at the first place					Conjunction at the second place					Remarks
	Normal Position	Top Position	Left-top position	Other special position	Unusual Shape	Right Position	Bottom Position	Right-Bottom position	Other special position	Unusual Shape	
ন	ন্য	ন্ড	ন্ট		ন্হ ন্ধ		গ্ন		ঘ্ন	হ্ন	ন- fala
প	প্য	প্ত	প্ট			স্প	ষ্প				
ফ	ফ্য					প্ফ					
ব	ব্ব	ব্জ	ব্দ		ব্ধ	ধ্ব	গ্ব		ক্ব		ব fala
ভ	ভ্য				ভ্র		দ্ভ				
ম	ম্য	ম্ন	ম্প			শ্ম			ক্ম	হ্ম	ম fala
য	য্ব								ক্য		্য- fala
র				র্ক					শ্র	ক্র ত্র ভ্র	Ref (র্) at the first position ্র fal at the second position
ল	ল্য	ল্ক	ল্ট			হ্ল	প্ল		ক্ল		ল- fala
শ	শ্য	শ্ন	শ্চ				শ্ম				
ষ	ষ্ব	ষ্ক	ষ্ঠ		ষ্ণ					ক্ষ	With ষ at second position one is of unusual shape and other with ref.
স	স্য	স্ক	স্থ		স্হ	ন্স				ক্স	
হ	হ্ব				হ্ন হ্ম	ল্হ					
য়											য় does not conjunct at the first or the second position
ড়											ড় does not conjunct at the first or the second position
ঢ়											ঢ় does not conjucted at the first or the second position
ৎ									র্ৎ		at the first position ৎ is written seperately at the second position conjuct only with ref
ং											at the first or second position ং is written seperately
ঃ											at the first or second position ঃ is written seperately

TABLE 3.2b CONJUNCTION ANALYSIS OF COMPOUND BYANJANA VARNAS WITH THREE BYANJANAS

VARNAS	CONJUNCTION AT THE FIRST PLACE WITH COMPOUND BYANJANA VARNAS WITH TWO BYANJANAS		CONJUNCTION AT THE LAST PLACE WITH COMPOUND BYANJANA VARNAS WITH TWO BYANJANAS		REMARKS
	Left-top position	Special position	Special position	Un usual shape	
ঙ	ঙ্ক্ষ				With ক্ষ only
ণ			ক্ষ্ণ		ণ fala with
ব			ক্ষ্ব		ব fala
ম			ক্ষ্ম		ম fala with ক্ষ
য			ক্ত্য		য fala
র		র্থ্য	ন্ত্র	ন্ত্র স্ত্র ষ্ট্র ঙ্ক্র ন্দ্র	ref(́)and র fala

TABLE 3.2c CONJUNCTION ANALYSIS OF COMPOUND BYANJANA VARNAS WITH FOUR BYANJANAS

Varnas	Conjuction at the last place with compound Byanjana varnas with three Byanjanas	Remarks
ব	র্দ্ধ্ব	ব-fala
য	র্দ্ধ্য	্য-fala

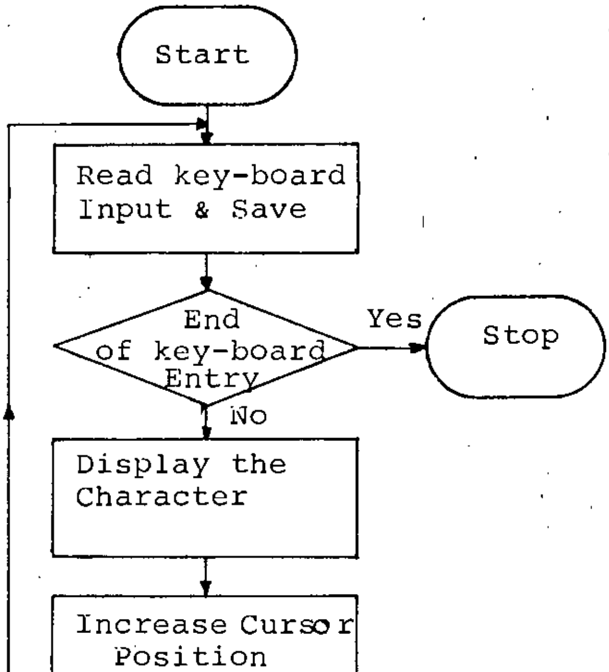


Fig. 3.1a: Algorithm of Mapping Swara Varnas, Byanjana Varnas, Aksharas with unusual shape of Swara-Kars, Numerals, Punctuation Marks and Special Graphic Symbols for Software Mapped Graphic Molecules.

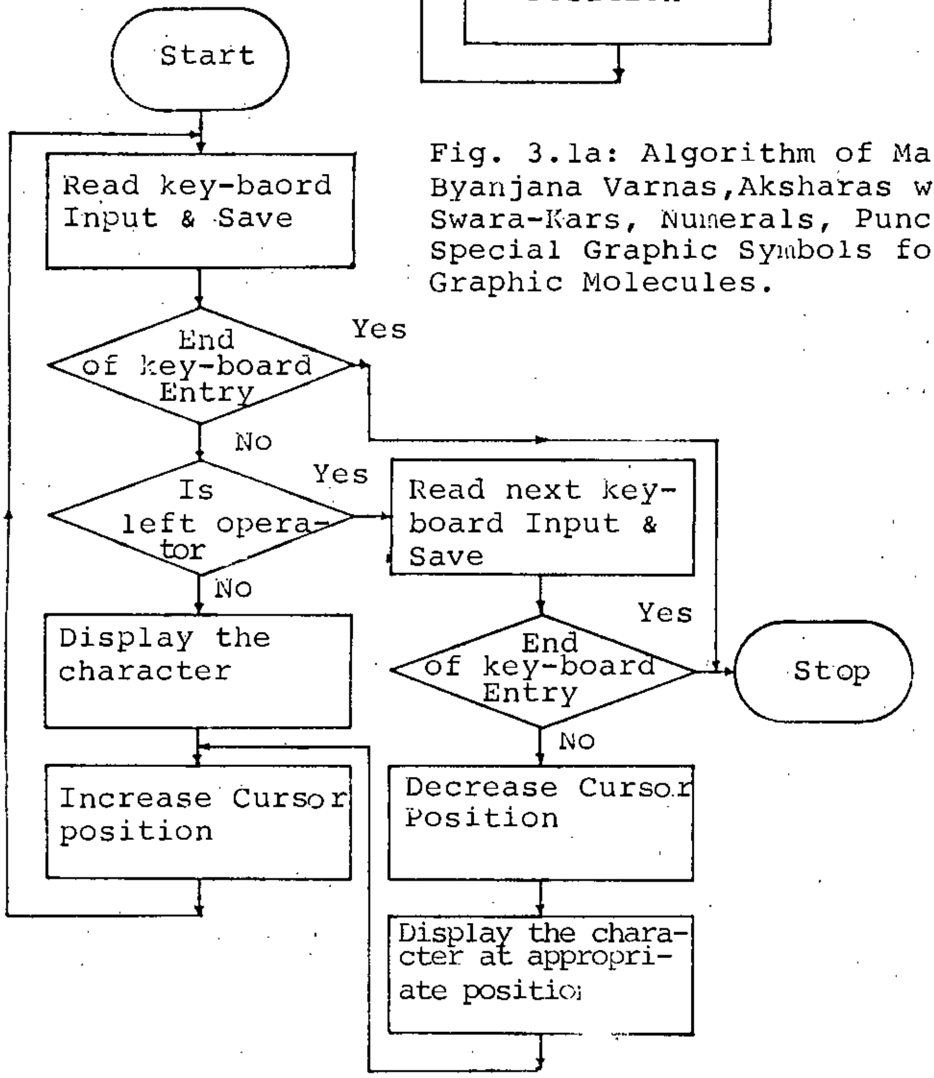


Fig. 3.1b: Algorithm of Using Diacritical Marks for Software Mapped Graphic Molecules.

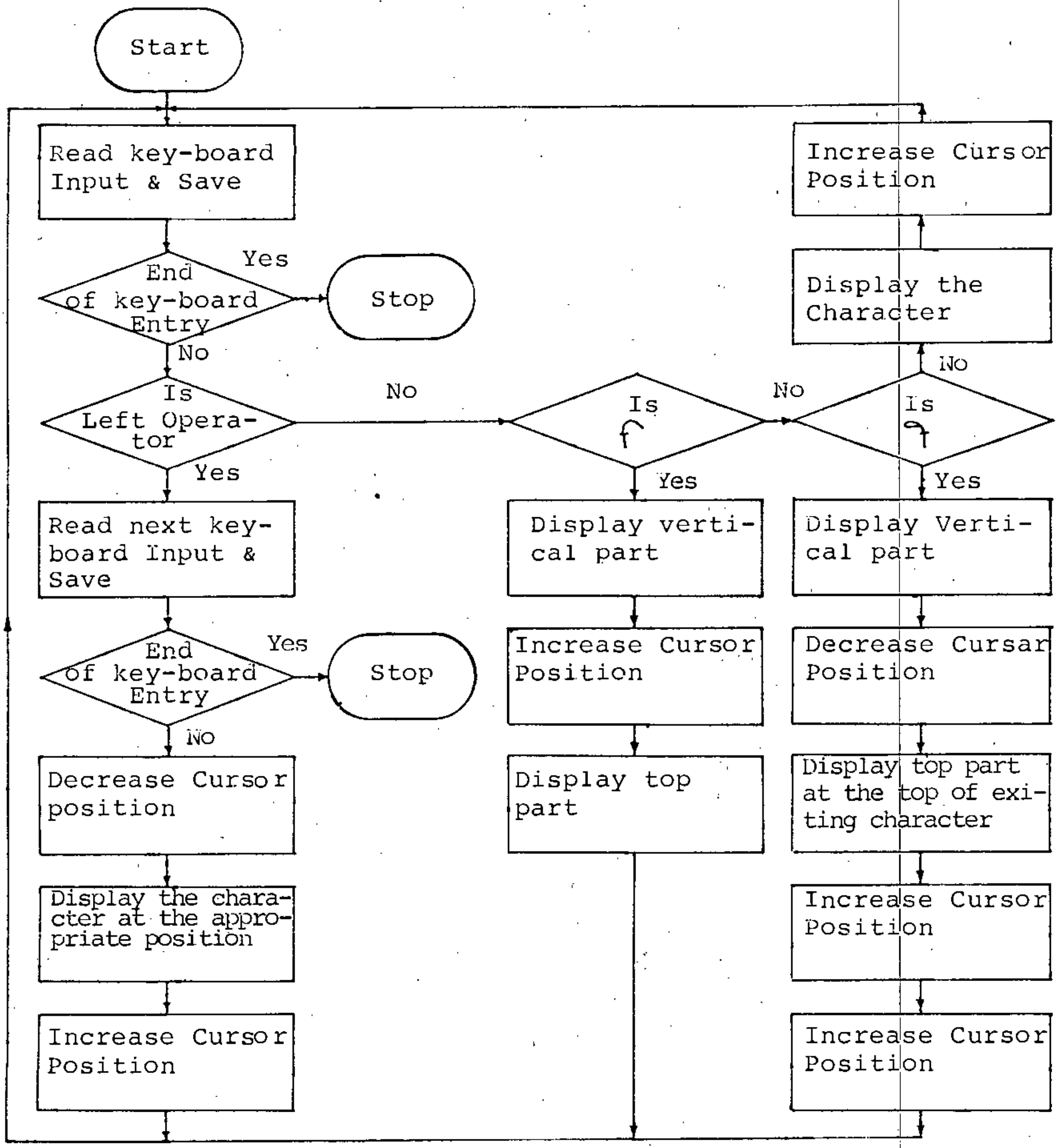


Fig. 3.1C : Algorithm of Using Swara-Kars for Software Mapped Graphic Molecules.

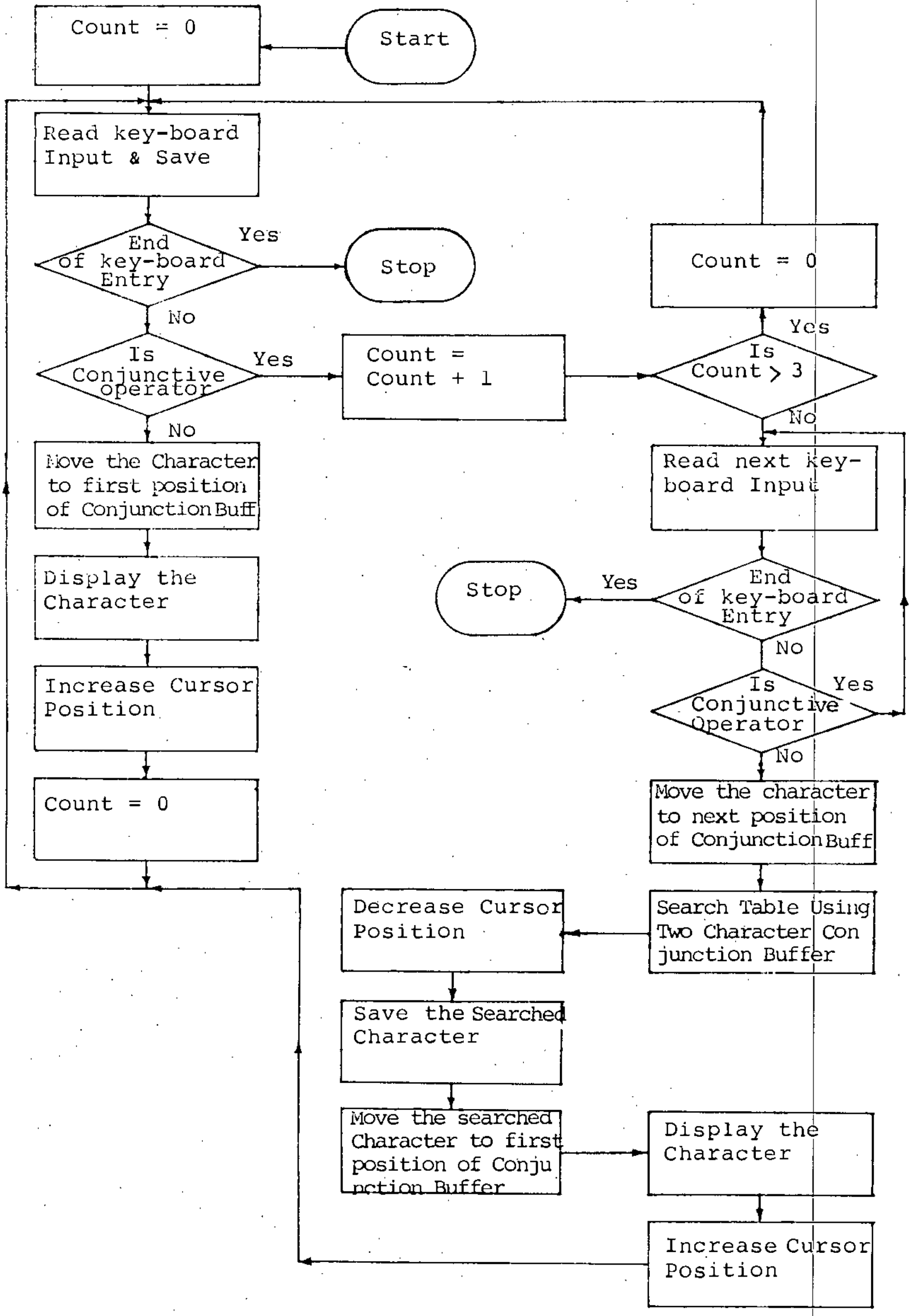
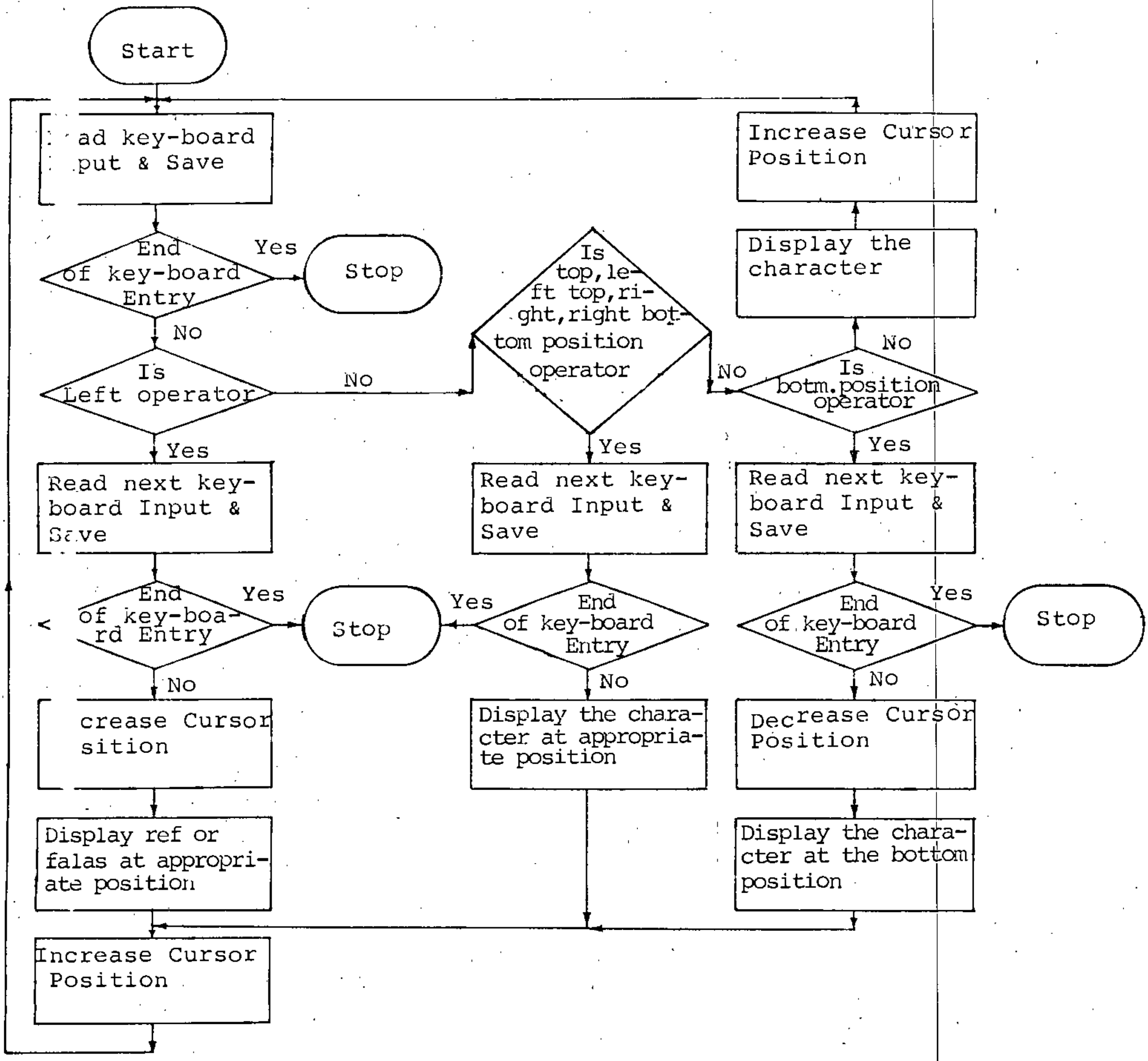


Fig. 3.1d: Algorithm of Mapping Compound Byanjana Varnas from Constituent Byanjana Varnas of Normal Shape and a Conjunctive Operator.



Fi 3.1e: Algorithm of Mapping Compound Byanjana Varnas from Constituent Byanjana Varnas of Normal Shape and Position Operators.

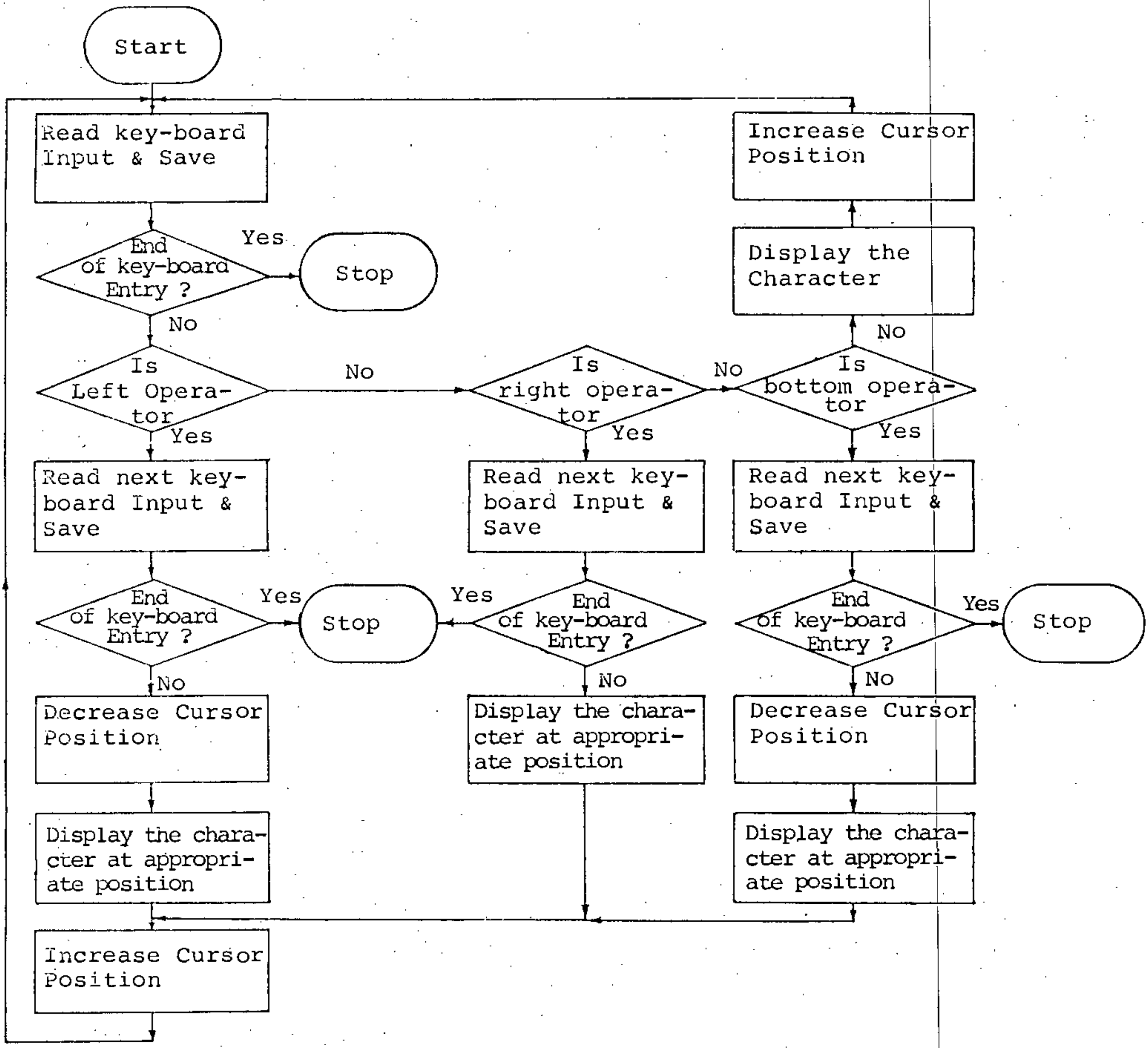


Fig. 3.1f. Algorithm of Mapping Compound Byanjana Varnas from Special Shaped Symbols and Conjunction Operators.

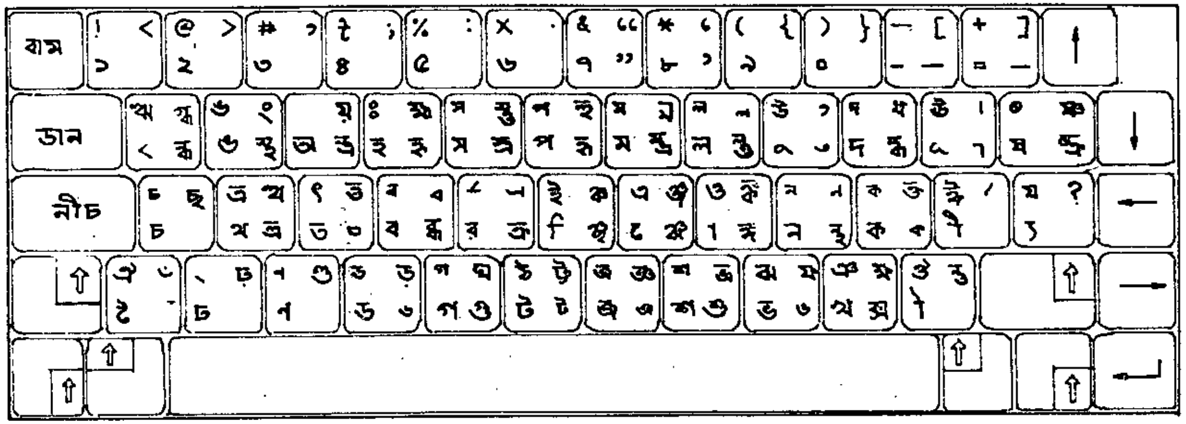


FIG. 3.2 KEY BOARD LAY-OUT OF KEY-BOARD PRIMITIVES FOR SOFTWARE MAPPED GRAPHIC MOLECULES

- Remembering the location of 172 primitives on a key-board will be difficult.
- The algorithm of implementing this method is highly complex and the speed of text entry, processing and displaying the text will be highly slow.
- This method can only be used in dot matrix printer. The 172 symbols can not be accomodated on a Daisy-wheel or a Line printer.
- Computer resource requirement and overhead will be considerably high in this method.
- This method can not be used in real-time applications.

3.4 KEY-BOARD PRIMITIVES SELECTION FOR DIRECTLY MAPPED GRAPHIC MOLECULES

To overcome the problems associated with the method of mapping graphic molecules under software control, a method of direct mapping the graphic molecules is required. In this direct mapping of graphic molecules, a set of graphic primitives are needed such that each and every graphic molecules can be generated by concatanating these graphic primitives, i.e., in this method, each and every graphic symbols are to be entered and displayed or printed seperately as a seperate entity and all graphic molecules are to be generated by these graphic symbols. The same set of graphic primitives are to be used as key-board primitives and the selection of these graphic symbols should be such that

- each symbol has its own lexical meaning such that, when represented internally in a computer system, lexical analysis can be done on the symbols
- all symbols can be encoded by a standard size of code bits
- all symbols can be accomodated on a standard size of key-board
- all graphic molecules can be directly mapped adequately by these symbols without the help of any software facilities such that the same Key-board can be used in an electronic type writer.

Selection of Bengali Graphic Primitives and their key-board lay-out are discussed in the following articles.

3.5 SELECTION OF BENGALI GRAPHIC PRIMITIVES

In the letter press printing of Bengali texts, no standard is being maintained about the shape of the Bengali compound byanjana varnas. The shape of these compound byanjana varnas varies widely from one letter press to another. On the other hand, the shape of these compound byanjana varnas of lino-type is completely different from that of letter press printing. For selecting the Bengali graphic primitives, need for the standardization of shape of the Bengali characters is felt. For selecting the graphic primitives for the present work, a shape for each character has been assumed and graphic primitives are selected based on that proposed shapes. Proposed graphics of all the Bengali characters of Table 2.1a to 2.1k, along with present lino-graphics, are given in Table 3.5.

The Swara Varans of Table 2.1a can be mapped by the graphic symbols of Table 3.3a. আ is to be mapped by concatenating অ and া of Table 3.3d. অ্যা or এ্যা are to be mapped by concatenating অ or এ , ্য of Table 3.3e and া of Table 3.3d.

The Byanjana Varnas of Table 2.1b can be mapped by the graphic symbols of Table 3.3b. The shape of ব and internal ব is identical and for that the internal ব is to be mapped by the ব.

The Diacritical Marks of Table 2.1c can be mapped by the graphic symbols of Table 3.3c. All these three diacritical marks are to be placed at the right of the concerned varnas.

The Swara-kars of Table 2.1d can be mapped by the graphic symbols of Table 3.3d. া , ী , ু , ূ , ৃ and ৗ are to be placed at the right and ি , ে and ৈ are to be placed at the left of the concerned byanjana varnas. ি and ী are not overlaped on the concerned byanjana varnas, ো is to be mapped by ে and া and ৌ is to be mapped by ে and ৗ.

Graphic symbols for mapping Compound Byanjana Varnas of Table 2.1e, 2.1f and 2.1g are selected based on the conjunction analysis of compound byanjana varnas of Table 3.2a, 3.2b and 3.2c. In the case of compound byanjana varnas with two byanjanas, for conjunction at the normal position, graphic symbols of Table 3.3b are adequately sufficient and the second varnas are to be placed as applicable as discussed latter. For conjunction at the top and left-top positions, 16 specially shaped small symbols of byanjana varnas are introduced which are to be placed at the left of the second varnas of normal shape and size. Among the 17 cases of conjunction at the top and left-top positions, 16 specially shaped small symbols of byanjana varnas are introduced and no such symbol is introduced for ট , because ট conjuncts in this position for only one case, i.e., ট্ঠ , which is not presently used in Bengali text and can be represented as ট্‌ঠ if needed. A symbol for ref (র্) is introduced which is to be placed at the right of the concerned varnas. For conjunction at the right, bottom and right-bottom positions, graphic symbols of Table 3.3b are adequately sufficient, which are to be placed at the right of the concerned specially shaped small byanjana varnas as applicable. For conjunction at other places, 7 fala symbols are introduced, which, except ্য , are to be placed at the right of the concerned byanjana varnas and the ্য is to be superposed under the concerned byanjana varnas as the aesthetics is highly hampered if ্র is placed at the right of the concerned varnas. In the case of ঢ্ব , ত্ব , ন্হ , হ্ব and হ্ল , all two byanjana varnas are of normal shape. In this case, specially shaped small varnas are used at the first position for ত্ব and ন্হ with normal varnas at the second position. For other cases fala symbols are used at the second position with normal varnas at the first position,as no specially shaped small symbols are introduced for ঢ and হ . Among the 26 compound byanjana varnas with two byanjanas and with unusual shape, 7 are kept in their unusual shapes and other 19 are proposed to write in their normal shapes, i.e., writing two byanjana varnas side by side or superposing ্র as applicable for other cases. ক্ষ and জ্ঞ are kept in their unusual shapes because their sound is completely deviated from the conjunct sound of their constituent varnas- ক্ষ sounds like খ and জ্ঞ sounds like গ্যঁ . ত্র and ক্ত are kept in their unusual shapes because no additional benifit is obtained from spliting their shapes. ত্ত ,হ্ম and ঙ্গ are kept in their unusual shapes because aesthetics is highly hampered on spliting their shapes. An additional specially shaped small symbol for ঞ is introduced to split the shape of all the unusual shaped compound byanjana varnas with ঞ at the first place. In the case of compound byanjana varnas with three byanjanas, specially shaped small symbol for ঙ , ref (র্), ব-fala, ম-fala, ন- fala, য- fala and ্র-fala are to be placed at the appropriate place with the appropriate compound byanjana varnas with two byanjanas of proposed shape as applicable. All 5 unusual shaped compound byanjana varnas with three byanjanas are to be written in their normal shapes, i.e., ্ is to be superposed under the concerned compound byanjana varnas with two byanjanas of proposed shape. In the case of compound byanjana varnas with four byanjanas, ্ব - fala and ্য- fala are to be placed at the right of the concerned compound byanjana varnas with three byanjanas of proposed shape. All these graphic symbols for mapping compound byanjana varnas are given in Table 3.3e.

All Aksharas with unusual shape of swara-kars of Table 2.1h are proposed to write in their normal shapes, i.e., ু , ূ and ৃ are to be placed at the right of the concerned byanjana varnas or compound byanjana varnas of proposed shape. So, no additional graphic symbol is needed for mapping aksharas with unusual shape of swara-kars.

The Numerals of Table 2.1i can be mapped by the graphic symbols of Table 3.3f.

The Punctuation Marks of Table 2.1j can be mapped by the graphic symbols of Table 3.3g. Hyphen (-) and dash (—) are to be mapped by the same symbol for hyphen (-). “ and ” are to be mapped by the same symbol ″ and ‘ and ’ are to be mapped by the same symbol ′ .

The Special Graphic Symbols of Table 2.1k can be mapped by the graphic symbols of Table 3.3h. Minus sign (-) is to be mapped by the hyphen symbol (-) of Table 3.3g.

All these 131 Bengali Graphic Symbols (BGS) of Table 3.3a to 3.3h are given in Table 3.4 in their lexical order. Mechanism of generation of all the 434 Bengali characters by the selected Bengali Graphic Symbols are given in Table 3.5 along with present Lino-graphics and proposed graphics. The algorithm of mapping all these Bengali characters is simple and given in Fig. 3.3.

3.6 STATISTICS OF THE BENGALI GRAPHIC PRIMITIVES

Prabir Kumar Das[1], in 1976, made a survey on the frequency of occurrence of Bengali characters on the basis of 43,126 no. of occurrence (appendix B). Another survey was made in 1984 by Gourhari Das et al.[4] on the frequency of occurrence of Extended Sen & Datta Graphic Symbol Set (SDBM) on the basis of 66,752 no. of occurrence (appendix A). During the present work, another survey was made on the frequency of occurrence of Bengali characters on the basis of 16,090 no. of occurrence from various recent periodicals covering poetry, general article, technical article, international affair and literature critisism etc. at random. This frequency of occurrence of Bengali characters is given in Table 3.6. Combining all these three survey reports (Table A-2, B-1 and 3.6), frequency of occurrence of the Bengali Graphic Symbols (BGS) has been computed on the basis of 140,688 no. of

TABLE 3.3a GRAPHIC SYMBOLS FOR MAPPING SWARA VARNAS

অ ই ঈ উ ঊ ঋ এ ঐ ও ঔ

TABLE 3.3b GRAPHIC SYMBOLS FOR MAPPING BYANJANA VARNAS

ক খ গ ঘ ঙ
চ ছ জ ঝ ঞ
ট ঠ ড ঢ ণ
ত থ দ ধ ন
প ফ ব ভ ম
য র ল শ
ষ স হ য় ড়
ঢ় ৎ ং ঃ

TABLE 3.3c GRAPHIC SYMBOLS FOR MAPPING DIACRITICAL MARKS

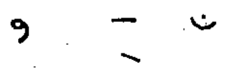


TABLE 3.3d GRAPHIC SYMBOLS FOR MAPPING SWARA–KARS

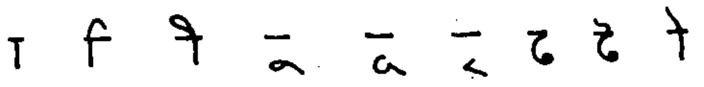


TABLE 3.3e GRAPHIC SYMBOLS FOR MAPPING COMPOUND BYANJANA VARNAS

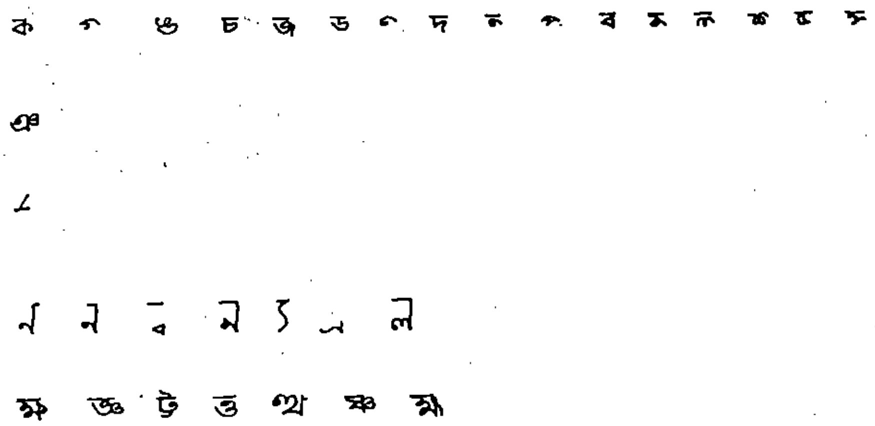


TABLE 3.3f GRAPHIC SYMBOLS FOR MAPPING NUMERALS

০ ১ ২ ৩ ৪ ৫ ৬ ৭ ৮ ৯

TABLE 3.3g GRAPHIC SYMBOLS FOR MAPPING PUNCTUATION MARKS

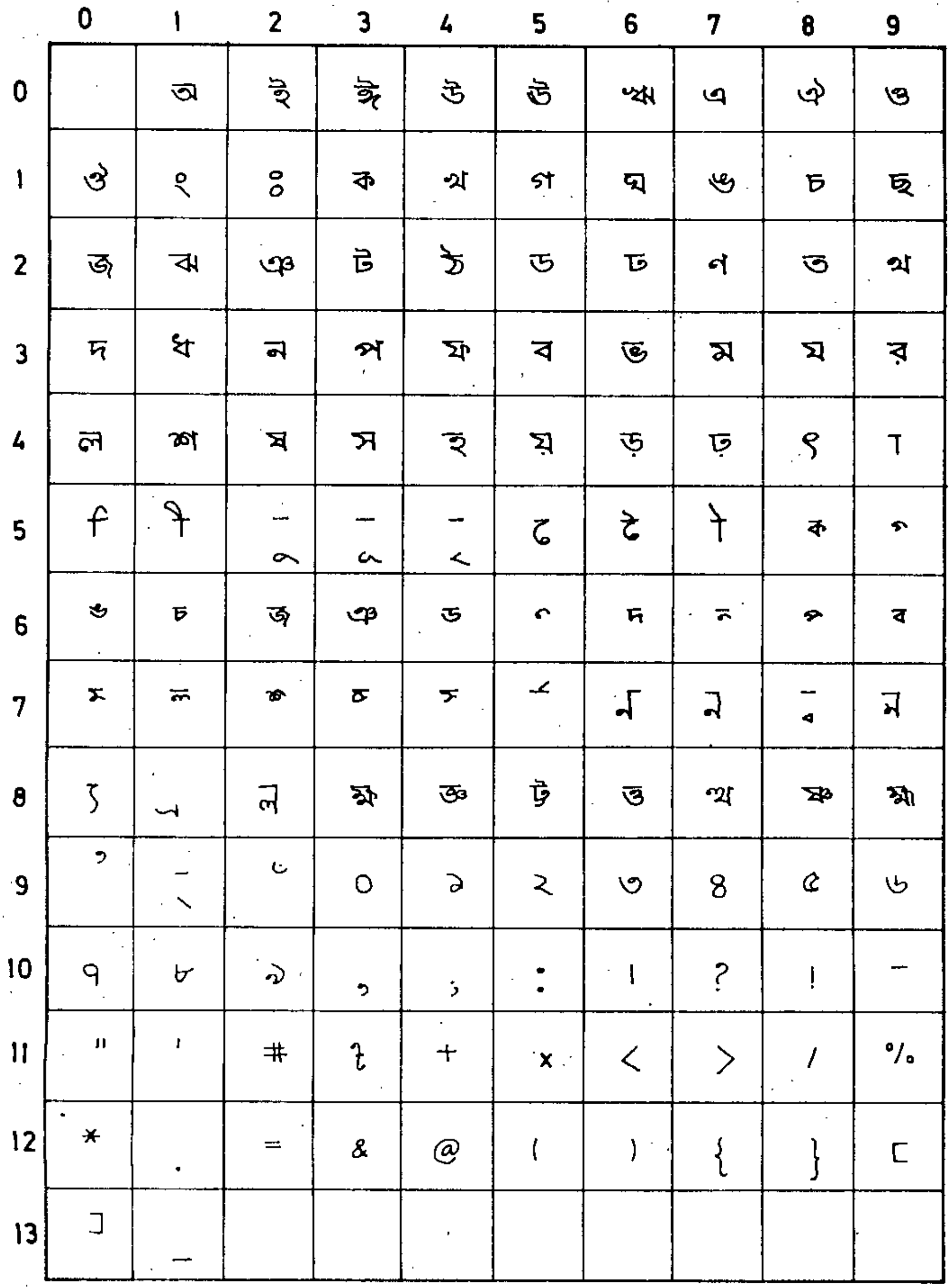
, ; : | ? ! — " '

TABLE 3.3h GRAPHIC SYMBOLS FOR MAPPING SPECIAL GRAPHIC SYMBOLS

₹ + × < > / % * . = & @ ( )

{ } [ ] _

TABLE 3.4 BENGALI GRAPHIC SYMBOL SET (BGS)



	0	1	2	3	4	5	6	7	8	9
0		অ	ই	ঈ	উ	ঊ	ঋ	এ	ঐ	ও
1	ঔ	ং	ঃ	ক	খ	গ	ঘ	ঙ	চ	ছ
2	জ	ঝ	ঞ	ট	ঠ	ড	ঢ	ণ	ত	থ
3	দ	ধ	ন	প	ফ	ব	ভ	ম	য	র
4	ল	শ	ষ	স	হ	য়	ড়	ঢ়	ৎ	া
5	ি	ী	ু	ূ	ৃ	ে	ৈ	ৗ	ক	গ
6	ঙ	চ	জ	ঞ	ড	ʼ	দ	ন	প	ব
7	ম	ল	শ	ষ	স	৴	ন	ন	ব	ম
8	ʃ	্র	ল্ল	ক্ষ	জ্ঞ	ট্র	ত্র	ঙ্খ	ষ্ক	ক্ষা
9	ʾ	্	ঁ	০	১	২	৩	৪	৫	৬
10	৭	৮	৯	,	;	:	।	?	!	-
11	"	'	#	ঽ	+	x	<	>	/	%
12	*	.	=	&	@	(	)	{	}	[
13	]	_								

(36)= ভ (127)= { etc.

TABLE 3.5 SCHEDULE OF BENGALI CHARACTERS, PRESENT LINO-GRAPHICS, PROPOSED GRAPHICS AND MECHANISM OF GENERATION BY BENGALI GRAPHIC SYMBOLS (BGS)

Characters	Present Lino-Graphics*	Proposed Graphics	Mechanism of generation by BGS**	Characters	Present Lino-Graphics*	Proposed Graphics	Mechanism of generation by BGS**
অ	অ	অ	(01)	ণ	ণ	ণ	(27)
আ	আ	আ	(01) (49)	ত	ত	ত	(28)
ই	ই	ই	(02)	থ	থ	থ	(29)
ঈ	ঈ	ঈ	(03)	দ	দ	দ	(30)
উ	উ	উ	(04)	ধ	ধ	ধ	(31)
ঊ	ঊ	ঊ	(05)	ন	ন	ন	(32)
ঋ	ঋ	ঋ	(06)	প	প	প	(33)
এ	এ	এ	(07)	ফ	ফ	ফ	(34)
ঐ	ঐ	ঐ	(08)	ব	ব	ব	(35)
ও	ও	ও	(09)	ভ	ভ	ভ	(36)
ঔ	ঔ	ঔ	(10)	ম	ম	ম	(37)
অ্যা/এ্যা	অ্যা/এ্যা	অ্যা/এ্যা	(01)+(80)+(49) (07)+(80)+(49)	য	য	য	(38)
ক	ক	ক	(13)	র	র	র	(39)
খ	খ	খ	(14)	ল	ল	ল	(40)
গ	গ	গ	(15)	ব		ব	(35)
ঘ	ঘ	ঘ	(16)	শ	শ	শ	(41)
ঙ	ঙ	ঙ	(17)	ষ	ষ	ষ	(42)
চ	চ	চ	(18)	স	স	স	(43)
ছ	ছ	ছ	(19)	হ	হ	হ	(44)
জ	জ	জ	(20)	য়	য়	য়	(45)
ঝ	ঝ	ঝ	(21)	ড়	ড়	ড়	(46)
ঞ	ঞ	ঞ	(22)	ঢ়	ঢ়	ঢ়	(47)
ট	ট	ট	(23)	ৎ	ৎ	ৎ	(48)
ঠ	ঠ	ঠ	(24)	ং	ং	ং	(11)
ড	ড	ড	(25)	ঃ	ঃ	ঃ	(12)
ঢ	ঢ	ঢ	(27)	ঁ	ঁ	ঁ	(90)

TABLE 3.5 (CONTINUED)

Characters	Present Lino-Graphics*	Proposed Graphics	Mechanism of generation by BGS**	Characters	Present Lino-Graphics*	Proposed Graphics	Mechanism of generation by BGS**
্	্	্	(91)	ঞ্জ	ঞ্জ	ঞজ	(63) + (20)
ঁ	ঁ	ঁ	(92)	ঞ্ঝ		ঞঝ	(63) + (21)
া	া	া	(49)	ট্ট	ট্ট	ট্ট	(85)
ি	ি	ি	(50)	ট্ঠ		ট্ ঠ	(23)+(91)+(24)
ী	ী	ী	(51)	ড্গ		ডগ	(64) + (15)
ু	ু	ু	(52)	ড্ড	ডড	ডড	(64) + (25)
ূ	ূ	ূ	(53)	ড্ঢ		ডঢ	(64) + (26)
ৃ	ৃ	ৃ	(54)	ণ্ট		ণট	(65) + (23)
ে	ে	ে	(55)	ণ্ঠ	ণঠ	ণঠ	(65) + (24)
ৈ	ৈ	ৈ	(56)	ণ্ড	ণড	ণড	(65) + (25)
ো	ো	ো	(55) + (49)	ণ্ঢ		ণঢ	(65) + (26)
ৌ	ৌ	ৌ	(55) + (57)	ত্ত	ত্ত	ত্ত	(86)
ক্ক	ক্ক	কক	(58) + (13)	ত্থ		ত্থ	(87)
ক্খ		কখ	(58) + (14)	দ্গ		দগ	(66) + (15)
ক্ট	ক্ট, কট	কট	(58) + (23)	দ্ঘ		দঘ	(66) + (16)
ক্ত	ক্ত, কত	কত	(58) + (28)	দ্দ	দদ	দদ	(66) + (30)
ক্ষ	ক্ষ	ক্ষ	(83)	দ্ধ	দধ	দধ	(66) + (31)
ক্স	ক্স	কস	(58) + (43)	দ্ভ	দভ	দভ	(66) + (36)
গ্গ		গগ	(59) + (15)	ন্ট	নট	নট	(67) + (23)
গ্ঘ		গঘ	(59) + (16)	ন্ঠ	নঠ	নঠ	(67) + (24)
গ্ধ		গধ	(59) + (31)	ন্ড	নড	নড	(67) + (25)
ঙ্ক	ঙ্ক, ঙক	ঙক	(60) + (13)	ন্ত	নত	নত	(67) + (28)
ঙ্খ	ঙখ	ঙখ	(60) + (14)	ন্থ	ন্থ, নথ	নথ	(67) + (29)
ঙ্গ	ঙগ	ঙগ	(60) + (15)	ন্দ	নদ	নদ	(67) + (30)
ঙ্ঘ	ঙঘ	ঙঘ	(60) + (16)	ন্ধ	নধ	নধ	(67) + (31)
চ্চ	চ্চ	চচ	(61) + (18)	ন্স	নস	নস	(67) + (43)
চ্ছ	চ্ছ	চছ	(61) + (19)	প্ট		পট	(68) + (23)
চ্ঞ		চঞ	(61) + (22)	প্ত	পত	পত	(68) + (28)
জ্জ	জজ	জজ	(62) + (20)	প্ন	পন	পন	(68) + (33)
জ্ঝ		জঝ	(62) + (21)	প্ফ		পফ	(68) + (34)
জ্ঞ	জ্ঞ	জ্ঞ	(84)	প্স		পস	(68) + (43)
ঞ্চ	ঞ্চ, ঞ্চ	ঞচ	(63) + (18)	ব্জ		বজ	(69) + (20)
ঞ্ছ	ঞ্ছ, ঞ্ছ	ঞছ	(63) + (19)	ব্দ	বদ	বদ	(69) + (30)

TABLE 3.5 (CONTINUED)

Characters	Present Lino-Graphics *	Proposed Graphics	Mechanism of generation by BGS **	Character-s	Present Lino-Graphics *	Proposed Graphics	Mechanism of generation by BGS **
ব্ধ	ব্ধ	ব্ধ	(69) + (31)	ঘ্ন		ঘ্ন	(16) + (77)
ব্ভ		ব্ভ	(69) + (36)	ণ্ন		ণ্ন	(65) + (32)
ম্প	ম্প	ম্প	(70) + (33)	হ্ণ		হ্ণ	(44) + (76)
ম্ফ		ম্ফ	(70) + (34)	ত্ন	ত্ন	ত্ন	(28) + (77)
ম্ভ	ম্ভ	ম্ভ	(70) + (36)	ধ্ন		ধ্ন	(31) + (77)
ল্ক	ল্ক	ল্ক	(71) + (13)	ন্ন	ন্ন	ন্ন	(67) + (32)
ল্গ		ল্গ	(71) + (15)	প্ন	প্ন	প্ন	(68) + (32)
ল্ট	ল্ট	ল্ট	(71) + (23)	ম্ন	ম্ন	ম্ন	(70) + (32)
ল্ড	ল্ড	ল্ড	(71) + (25)	শ্ন	শ্ন	শ্ন	(72) + (32)
ল্প	ল্প	ল্প	(71) + (33)	স্ন	স্ন	স্ন	(74) + (32)
ল্ফ	ল্ফ	ল্ফ	(71) + (34)	হ্ন	হ্ন	হ্ন	(44) + (77)
ল্ভ		ল্ভ	(71) + (36)	ক্ব		ক্ব	(13) + (78)
ল্হ		ল্হ	(71) + (44)	গ্ব		গ্ব	(59) + (35)
শ্চ	শ্চ	শ্চ	(72) + (18)	জ্ব	জ্ব	জ্ব	(20) + (78)
শ্ছ		শ্ছ	(72) + (19)	ট্ব		ট্ব	(23) + (78)
শ্ল		শ্ল	(72) + (41)	ড্ব		ড্ব	(25) + (78)
ষ্ক	ষ্ক	ষ্ক	(73) + (13)	ণ্ব	ণ্ব	ণ্ব	(65) + (35)
ষ্ক্র		ষ্ক্র	(73) + (22)	ত্ব	ত্ব	ত্ব	(28) + (78)
ষ্ট	ষ্ট	ষ্ট	(73) + (23)	থ্ব		থ্ব	(29) + (78)
ষ্ঠ	ষ্ঠ	ষ্ঠ	(73) + (24)	দ্ব	দ্ব, দ্ব	দ্ব	(66) + (35)
ষ্ণ	ষ্ণ	ষ্ণ	(88)	ধ্ব		ধ্ব	(31) + (78)
ষ্প	ষ্প	ষ্প	(73) + (33)	ন্ব	ন্ব	ন্ব	(67) + (35)
ষ্ফ		ষ্ফ	(73) + (34)	ব্ব	ব্ব	ব্ব	(69) + (35)
স্ক	স্ক	স্ক	(74) + (13)	ম্ব	ম্ব	ম্ব	(70) + (35)
স্খ		স্খ	(74) + (14)	য্ব		য্ব	(38) + (78)
স্ট	স্ট	স্ট	(74) + (23)	ল্ব		ল্ব	(71) + (35)
স্ত	স্ত	স্ত	(74) + (28)	শ্ব	শ্ব	শ্ব	(72) + (35)
স্থ	স্হ, স্থ	স্থ	(74) + (29)	ষ্ব		ষ্ব	(42) + (78)
স্প	স্প	স্প	(74) + (33)	স্ব	স্ব	স্ব	(74) + (35)
স্ফ		স্ফ	(74) + (34)	হ্ব	হ্ব	হ্ব	(44) + (78)
হ্ম	হ্ম	হ্ম	(89)	ক্ম		ক্ম	(13) + (79)
গ্ণ		গ্ণ	(59) + (27)	গ্ম	গ্ম	গ্ম	(15) + (79)
ঘ্ণ		ঘ্ণ	(16) + (76)	ঙ্ম		ঙ্ম	(17) + (79)
ণ্ণ		ণ্ণ	(65) + (27)	জ্ম		জ্ম	(20) + (79)
গ্ন	গ্ন	গ্ন	(59) + (32)	ট্ম		ট্ম	(23) + (79)

Characters	Present Lino-Graphics*	Proposed Graphics	Mechanism of generation by BGS**	Characters	Present Lino-Graphics*	Proposed Graphics	Mechanism of generation by BGS**
ড্ম		ড্ম	(25) + (79)	ম্য	ম্য	ম্য	(37) + (80)
ণ্ম		ণ্ম	(65) + (37)	য্য	য্য	য্য	(38) + (80)
ত্ম	ত্ম	ত্ম	(28) + (79)	ল্য	ল্য	ল্য	(40) + (80)
দ্ম	দ্ম	দ্ম	(30) + (79)	শ্য	শ্য	শ্য	(41) + (80)
ধ্ম		ধ্ম	(31) + (79)	ষ্য	ষ্য	ষ্য	(42) + (80)
ন্ম	ন্ম	ন্ম	(67) + (37)	স্য	স্য	স্য	(43) + (80)
প্ম		প্ম	(33) + (79)	হ্য		হ্য	(44) + (80)
ম্ম	ম্ম	ম্ম	(70) + (37)	ক্র	ক্র	ক্র	(13)/(81)
ল্ম		ল্ম	(71) + (37)	খ্র		খ্র	(14)/(81)
শ্ম		শ্ম	(41) + (79)	গ্র	গ্র	গ্র	(15) / (81)
ষ্ম		ষ্ম	(73) + (37)	ঘ্র		ঘ্র	(16) / (81)
স্ম	স্ম	স্ম	(74) + (37)	জ্র	জ্র	জ্র	(20) / (81)
ক্য	ক্য	ক্য	(13) + (80)	ট্র	ট্র	ট্র	(23) / (81)
খ্য	খ্য	খ্য	(14) + (80)	ড্র	ড্র	ড্র	(25) / (81)
গ্য	গ্য	গ্য	(15) + (80)	ত্র	ত্র	ত্র	(28)/(81)
ঘ্য		ঘ্য	(16) + (80)	থ্র		থ্র	(29)/(81)
চ্য	চ্য	চ্য	(18) + (80)	দ্র	দ্র	দ্র	(30)/(81)
ছ্য		ছ্য	(19) + (80)	ধ্র		ধ্র	(31) / (81)
জ্য	জ্য	জ্য	(20) + (80)	প্র	প্র	প্র	(33) / (81)
ট্য	ট্য	ট্য	(23) + (80)	ফ্র	ফ্র	ফ্র	(34) / (81)
ঠ্য		ঠ্য	(24) + (80)	ব্র	ব্র	ব্র	(35) / (81)
ড্য		ড্য	(25) + (80)	ভ্র	ভ্র	ভ্র	(36) / (81)
ঢ্য		ঢ্য	(26) + (80)	ম্র		ম্র	(37) / (81)
ণ্য		ণ্য	(27) + (80)	শ্র	শ্র	শ্র	(41) / (81)
ত্য	ত্য	ত্য	(28) + (80)	স্র	স্র	স্র	(43) / (81)
থ্য	থ্য	থ্য	(29) + (80)	হ্র		হ্র	(44) / (81)
দ্য	দ্য	দ্য	(30) + (80)	ক্ল	ক্ল	ক্ল	(13) + (82)
ধ্য	ধ্য	ধ্য	(31) + (80)	গ্ল		গ্ল	(59) + (40)
ন্য	ন্য	ন্য	(32) + (80)	প্ল	প্ল	প্ল	(68) + (40)
প্য	প্য	প্য	(33) + (80)	ফ্ল		ফ্ল	(34) + (82)
ফ্য	ফ্য	ফ্য	(34) + (80)	ব্ল		ব্ল	(35) + (82)
ব্য	ব্য	ব্য	(35) + (80)	ভ্ল		ভ্ল	(36) + (82)
ভ্য	ভ্য	ভ্য	(36) + (80)	ম্ল	ম্ল	ম্ল	(70) + (40)

TABLE 3.5 (CONTINUED)

Characters	Present Lino-Graphics*	Proposed Graphics	Mechanism of generation by BGS**	Characters	Present Lino-Graphics*	Proposed Graphics	Mechanism of generation by BGS**
ল্ল	ল্ল	ল্ল	(71) + (40)	ক্ষ্ণ		ক্ষ্ণ	(83) + (76)
শ্ল	শ্ল	শ্ল	(72) + (40)				
স্ল		স্ল	(74) + (40)	ক্ষ্ম	ক্ষ্ম	ক্ষ্ম	(83) + (79)
হ্ল		হ্ল	(44) + (82)	ক্র্য	ক্র্য	ক্র্য	(13)/(81) + (80)
র্ক	র্ক	র্ক	(13) + (75)	ক্ল্য		ক্ল্য	(13) + (82) + (80)
র্খ		র্খ	(14) + (75)	ক্ষ্য	ক্ষ্য	ক্ষ্য	(83) + (80)
র্গ	র্গ	র্গ	(15) + (75)	গ্ন্য		গ্ন্য	(59) + (32) + (80
র্ঘ	র্ঘ	র্ঘ	(16) + (75)	গ্র্য		গ্র্য	(15)/(81) + (80)
র্চ	র্চ	র্চ	(18) + (75)	ঙ্ক্য		ঙ্ক্য	(60)+(13)+(80)
র্ছ		র্ছ	(19) + (75)	ঙ্গ্য		ঙ্গ্য	(60) + (15)+(80)
র্জ	র্জ	র্জ	(20) + (75)	ঙ্ঘ্য		ঙ্ঘ্য	(60) + (16)+(60)
র্ঝ		র্ঝ	(21) + (75)	ণ্ঠ্য		ণ্ঠ্য	(65) + (24) + (80)
র্ট	র্ট	র্ট	(23) + (75)	ণ্ড্য		ণ্ড্য	(65) + (25)+(80)
র্ড	র্ড	র্ড	(25) + (75)	ত্ম্য		ত্ম্য	(28) + (79)+(80)
র্ণ	র্ণ	র্ণ	(27) + (75)	ত্র্য		ত্র্য	(28)/(81) + (80)
র্ত	র্ত	র্ত	(28) + (75)	দ্ব্য		দ্ব্য	(66) + (35) + (80)
র্থ	র্থ	র্থ	(29) + (75)	ন্ত্য		ন্ত্য	(67)+ (28)+(80)
র্দ	র্দ	র্দ	(30) + (75)	ন্দ্য		ন্দ্য	(67) + (30)+(80)
র্ধ	র্ধ	র্ধ	(31) + (75)	ন্ধ্য	ন্ধ্য	ন্ধ্য	(67) + (31) + (80)
র্ন	র্ন	র্ন	(32) + (75)	ন্ন্য		ন্ন্য	(67) + (32)+(80)
র্প	র্প	র্প	(33) + (75)	প্ল্য	প্ল্য	প্ল্য	(68) + (40) + (80)
র্ফ	র্ফ	র্ফ	(34) + (75)	ষ্ক্য		ষ্ক্য	(88) + (80)
র্ব	র্ব	র্ব	(35) + (75)	ষ্ট্য		ষ্ট্য	(73)+(23) + (80)
র্ভ	র্ভ	র্ভ	(36) + (75)	ষ্ঠ্য		ষ্ঠ্য	(73) + (24) + (80)
র্ম	র্ম	র্ম	(37) + (75)	ষ্ম্য		ষ্ম্য	(73) + (37)+(80)
র্য	র্য	র্য	(38) + (75)	স্ট্য	স্ট্য	স্ট্য	(74) + (23)+(80)
র্ল	র্ল	র্ল	(40) + (75)	স্ত্য		স্ত্য	(74)+ (28)+(80)
র্শ	র্শ	র্শ	(41) + (75)	ক্ত্র		ক্ত্র	(58)+ (28)/(81)
র্ষ	র্ষ	র্ষ	(42) + (75)	ঙ্ক্র		ঙ্ক্র	(60)+(13)/(81)
র্স	র্স	র্স	(43) + (75)	ঙ্খ্র		ঙ্খ্র	(60)+(14)/(81)
র্হ		র্হ	(44) + (75)	ঙ্ঘ্র		ঙ্ঘ্র	(60) + (16)/(81)
র্ৎ		র্ৎ	(48) + (75)	চ্ছ্র		চ্ছ্র	(61) + (19)/(81)
ঙ্ক্ষ		ঙ্ক্ষ	(60) + (83)	ণ্ড্র		ণ্ড্র	(65) + (25)/(81)

TABLE 3.5 (CONTINUED)

Characters	Present Lino-Graphics *	Proposed Graphics	Mechanism of generation by BGS **	Characters	Present Lino-Graphics *	Proposed Graphics	Mechanism of generation by BGS **
ত্র		ত্র	(86)/(81)	র্ব্ভ		র্ব্ভ	(69)+ (36)+ (75)
ন্ত্র	ন্ত্র	ন্ত্র	(67)+(28)/(81)	র্ব্য		র্ব্য	(35) + (75) + (80)
ন্দ্র	ন্দ্র	ন্দ্র	(67)+(30)/(81)	র্ম্ম		র্ম্ম	(70) + (37) + (75)
ন্ধ্র		ন্ধ্র	(67)+(31)/(81)	র্য্য	র্য্য	র্য্য	(38) + (75) + (80)
ম্প্র	ম্প্র	ম্প্র	(70)+(33)/(81)	র্শ্ব	র্শ্ব	র্শ্ব	(72) + (35) + (75)
ম্ভ্র		ম্ভ্র	(70)+(36)/(81)	র্ষ্ণ		র্ষ্ণ	(88) + (75)
ষ্ক্র		ষ্ক্র	(73)+(13)/(81)	র্ষ্য		র্ষ্য	(42) + (75)+(80)
ষ্ট্র	ষ্ট্র	ষ্ট্র	(73)+(23)/(81)	র্দ্দ্য		র্দ্দ্য	(66)+(30)+ (75)+(80)
ষ্প্র		ষ্প্র	(73)+(33)/(81)	র্দ্ধ্ব		র্দ্ধ্ব	(66)+(31)+(78) +(75
স্ট্র	স্ট্র	স্ট্র	(74)+(23)/(81)	র্ম্ম্য		র্ম্ম্য	(70)+ (37)+(75)+(80
স্ত্র	স্ত্র	স্ত্র	(74)+(28)/(81)	ক্ষ্ম্য		ক্ষ্ম্য	(83)+(79)+(80)
স্প্র		স্প্র	(74) + (33)/(81)	ন্ত্র্য		ন্ত্র্য	(67)+(28)/(81)+(80)
ক্ষ্ব		ক্ষ্ব	(83)+(78)	ঙ্ক্ষ্য		ঙ্ক্ষ্য	(60)+(83)+(80)
চ্ছ্ব		চ্ছ্ব	(61)+(19)+(78)	গু	গু	গু	(15)+(52)
জ্জ্ব	জ্জ্ব	জ্জ্ব	(62)+(20)+(78)	রু	রু	রু	(39) + (52)
ত্ত্ব	ত্ত্ব	ত্ত্ব	(86)+(78)	রূ	রূ	রূ	(39)+(53)
ন্ত্ব		ন্ত্ব	(67)+(28)+(78)	শু	শু	শু	(41) + (52)
ন্দ্ব	ন্দ্ব	ন্দ্ব	(67)+(30)+(78)	হু	হু	হু	(44) + (52)
র্ঘ্য		র্ঘ্য	(16)+ (75)+(80)	হৃ	হৃ	হৃ	(44) + (54)
র্চ্চ		র্চ্চ	(61) + (18)+(75)	ন্তু	ন্তু, ন্তু	ন্তু	(67) + (28)+(52)
র্চ্ছ		র্চ্ছ	(61) + (19)+(75)	ঙ্গু		ঙ্গু	(71) + (15)+(52)
র্জ্জ		র্জ্জ	(62) + (20)+(75)	স্তু	স্তু, স্তু	স্তু	(74)+ (28)+(52)
র্ট্য		র্ট্য	(26)+ (75)+ (80)	ত্রু	ত্রু	ত্রু	(28)/(81) + (52)
র্ণ্য		র্ণ্য	(27)+ (75)+(80)	ত্রূ		ত্রূ	(28)/(81) + (53)
র্ত্ত		র্ত্ত	(86) + (75)	দ্রু		দ্রু	(30)/(81) + (52)
র্ত্য		র্ত্য	(28) + (75)+(80)	দ্রূ		দ্রূ	(30)/(81)+(53)
র্থ্য		র্থ্য	(29)+(75) +(80)	ধ্রু		ধ্রু	(31)/(81) + (52)
র্দ্দ		র্দ্দ	(66) + (30) +(75)	ধ্রূ		ধ্রূ	(31)/(81) + (53)
র্দ্ধ		র্দ্ধ	(66)+(31)+(75)	প্রু		প্রু	(33)/(81) + (52)
র্দ্ব		র্দ্ব	(66)+(35)+(75)	প্রূ		প্রূ	(33)/(81) + (53)
র্দ্য		র্দ্য	(30)+ (75)+(80)	ব্রু		ব্রু	(35)/(81) + (52)
র্দ্র	র্দ্র	র্দ্র	(30)/ (81)+(75)	ব্রূ		ব্রূ	(35)/(81) + (53)
র্ধ্ব		র্ধ্ব	(31)+ (78)+(75)	ভ্রু		ভ্রু	(36)/(81)+(52)
র্ব্ব		র্ব্ব	(69) + (35) + (75)	ভ্রূ		ভ্রূ	(36)/(81)+(53)

TABLE 3.5 (CONTINUED)

Characters	Present Lino-Graphics*	Proposed Graphics	Mechanism of generation by BGS**	Characters	Present Lino-Graphics*	Proposed Graphics	Mechanism of generation by BGS**
শ্রু	শ্রু	শ্রু	(41)/(81)+(52)	%	%	%	(119)
শ্রূ		শ্রূ	(41)/(81)+(53)	*		*	(120)
স্রু		স্রু	(43)/(81)+(52)	.	.	.	(121)
স্রূ		স্রূ	(43)/(81)+(53)	=		=	(122)
০	০	০	(93)	&		&	(123)
১	১	১	(94)	@		@	(124)
২	২	২	(95)	(	(	(	(125)
৩	৩	৩	(96)	)	)	)	(126)
৪	৪	৪	(97)	{		{	(127)
৫	৫	৫	(98)	}		}	(128)
৬	৬	৬	(99)	[		[	(129)
৭	৭	৭	(100)	]		]	(130)
৮	৮	৮	(101)	_		_	(131)
৯	৯	৯	(102)				
,	,	,	(103)				
;	;	;	(104)				
:	:	:	(105)				
।	।	।	(106)				
?	?	?	(107)				
!	!	!	(108)				
-	-	-	(109)				
—	—	-	(109)				
“		"	(110)				
”		"	(110)				
‘	‘	'	(111)				
’	’	'	(111)				
#		#	(112)				
৳		৳	(113)				
+		+	(114)				
-		-	(109)				
×		×	(115)				
<		<	(116)				
>		>	(117)				
/		/	(118)				

* Note: 1. An empty entry indicates no character found during survey.

2. More than one entry means more than one graphics found.

** Note: 1. ( ) means graphic symbol of Table 3.4 corresponding to number enclosed.

2. + means concatenation.

3. / means superposition.

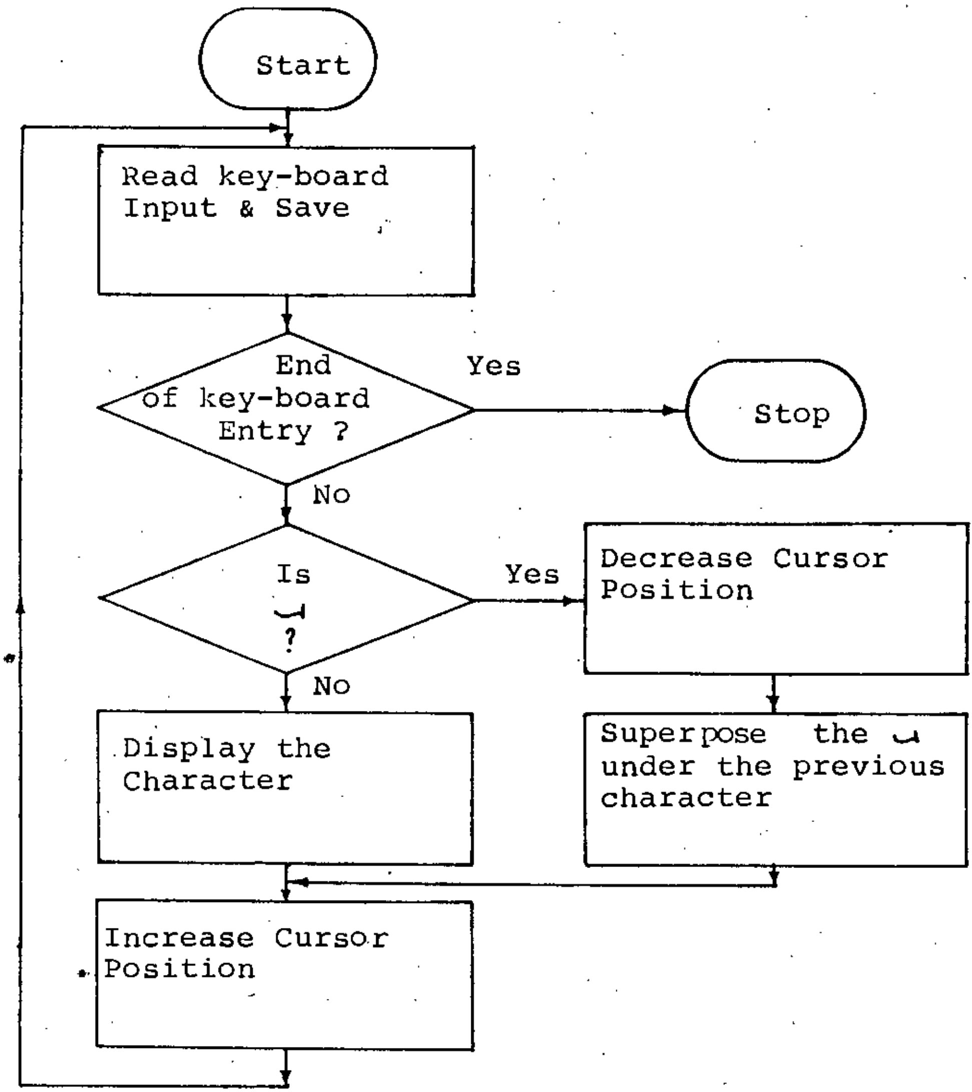


Fig. 3.3: Algorithm of Mapping Bengali Characters by the Bengali Graphic Symbols.

Table 3.6: Frequency of Occurrence (On the Basis of 16,090 No. of Occurrence) of Bengali Characters

Character Seq. No.	No. of occurrence	% of Occurrence	Character Seq. No.	No. of Occurrence	% of Occurrence
(001)	114	0.7085	(024)	16	0.0994
(002)	109	0.6774	(025)	37	0.2299
(003)	160	0.9944	(026)	3	0.0186
(004)	0	0.0000	(027)	59	0.3666
(005)	55	0.3418	(028)	387	2.4052
(006)	0	0.0000	(029)	87	0.5407
(007)	0	0.0000	(030)	256	1.5910
(008)	186	1.1559	(031)	54	0.3356
(009)	3	0.0186			
(010)	110	0.6836	(032)	606	3.7663
(011)	1	0.0062	(033)	246	1.5288
(012)	2	0.0124	(034)	26	0.1615
(013)	505	3.1385	(035)	453	2.8154
(014)	77	0.4785	(036)	96	0.5966
(015)	112	0.6960	(037)	343	2.1317
(016)	27	0.1678	(038)	79	0.4909
(017)	5	0.0310	(039)	853	5.3014
(018)	72	0.4474	(040)	360	2.2374
(019)	160	0.9944	(041)	0	0.0000
(020)	125	0.7768	(042)	135	0.8390
(021)	16	0.0994	(043)	23	0.1429
(022)	0	0.0000	(044)	391	2.4300
(023)	146	0.9073	(045)	185	1.1497

Table 3.6 Contd.

Character Seq. No.	No. of Occurrence	% of Occurrence	Character Seq. No.	No. of Occurrence	% of Occurrence
(046)	249	1.5475	(068)	17	0.1056
(047)	62	0.3853	(069)	0	0.0000
(048)	1	0.0062	(070)	0	0.0000
(049)	41	0.2548	(071)	1	0.0062
(050)	62	0.3853	(072)	8	0.0497
(051)	12	0.0745	(073)	1	0.0062
(052)	10	0.0621	(074)	45	0.2796
(053)	1	0.0062	(075)	2	0.0124
(054)	32	0.1988	(076)	5	0.0310
(055)	1,366	8.4897	(077)	0	0.0000
(056)	732	4.5494	(078)	0	0.0000
(057)	133	0.8266	(079)	0	0.0000
(058)	217	1.3486	(080)	0	0.0000
(059)	43	0.2672	(081)	0	0.0000
(060)	36	0.2237	(082)	1	0.0062
(061)	1,281	7.9614	(083)	0	0.0000
(062)	20	0.1243	(084)	1	0.0062
(063)	214	1.3300	(085)	2	0.0124
(064)	6	0.0372	(086)	33	0.2050
(065)	2	0.0124	(087)	0	0.0000
(066)	0	0.0000	(088)	0	0.0000
(067)	2	0.0124	(089)	0	0.0000

Table 3.6 Contd.

Character Seq. No.	No. of Occurrence	% of Occurrence	Character Seq. No.	No. of Occurrence	% of Occurrence
(090)	0	0.0000	(111)	9	0.0559
(091)	0	0.0000	(112)	0	0.0000
(092)	11	0.0683	(113)	1	0.0062
(093)	2	0.0124	(114)	0	0.0000
(094)	26	0.1615	(115)	0	0.0000
(095)	1	0.0062	(116)	0	0.0000
(096)	0	0.0000	(117)	1	0.0062
(097)	3	0.0186	(118)	1	0.0062
(098)	19	0.1180	(119)	0	0.0000
(099)	0	0.0000	(120)	0	0.0000
(100)	1	0.0062	(121)	1	0.0062
(101)	0	0.0000	(122)	0	0.0000
(102)	2	0.0124	(123)	0	0.0000
(103)	0	0.0000	(124)	0	0.0000
(104)	3	0.0186	(125)	0	0.0000
(105)	2	0.0124	(126)	1	0.0062
(106)	0	0.0000	(127)	0	0.0000
(107)	18	0.1118	(128)	0	0.0000
(108)	1	0.0062	(129)	1	0.0062
(109)	32	0.1988	(130)	9	0.0559
(110)	1	0.0062	(131)	0	0.0000

Table 3.6 Contd.

Character Seq. No.	No. of Occurrence	% of Occurrence	Character Seq. No.	No. of Occurrence	% of Occurrence
(132)	0	0.0000	(152)	1	0.0062
(133)	0	0.0000	(153)	1	0.0062
(134)	1	0.0062	(154)	22	0.1367
(135)	0	0.0000	(155)	5	0.0310
(136)	0	0.0000	(156)	0	0.0000
(137)	13	0.0807	(157)	0	0.0000
(138)	0	0.0000	(158)	0	0.0000
(139)	2	0.0124	(159)	13	0.0807
(140)	7	0.0435	(160)	0	0.0000
(141)	2	0.0124	(161)	13	0.0807
(142)	22	0.1367	(162)	1	0.0062
(143)	19	0.1180	(163)	4	0.0248
(144)	0	0.0000	(164)	16	0.0994
(145)	3	0.0186	(165)	4	0.0248
(146)	0	0.0000	(166)	10	0.0621
(147)	0	0.0000	(167)	26	0.1615
(148)	0	0.0000	(168)	12	0.0745
(149)	2	0.0124	(169)	1	0.0062
(150)	19	0.1180	(170)	4	0.0248
(151)	2	0.0124	(171)	32	0.1988

Table 3.6 Contd.

Character Seq. No.	No. of Occurrence	% of Occurrence	Character Seq. No.	No. of Occurrence	% of Occurrence
(172)	2	0.0124	(195)	1	0.0062
(173)	0	0.0000	(196)	0	0.0000
(174)	2	0.0124	(197)	1	0.0062
(175)	2	0.0124	(198)	16	0.0994
(176)	1	0.0062	(199)	0	0.0000
(177)	0	0.0000	(200)	10	0.0621
(178)	0	0.0000	(201)	3	0.0186
(179)	1	0.0062	(202)	5	0.0310
(180)	77	0.4785	(203)	12	0.0745
(181)	1	0.0062	(204)	0	0.0000
(182)	0	0.0000	(205)	2	0.0124
(183)	1	0.0062	(206)	0	0.0000
(184)	16	0.0994	(207)	1	0.0062
(185)	0	0.0000	(208)	8	0.0497
(186)	0	0.0000	(209)	0	0.0000
(187)	1	0.0062	(210)	7	0.0435
(188)	1	0.0062	(211)	1	0.0062
(189)	1	0.0062	(212)	1	0.0062
(190)	0	0.0000	(213)	0	0.0000
(191)	13	0.0807	(214)	6	0.0372
(192)	2	0.0124	(215)	0	0.0000
(193)	0	0.0000	(216)	9	0.0559
(194)	1	0.0062	(217)	1	0.0062

Table 3.6 Contd.

Character Seq. No.	No. of Occurrence	% of Occurrence	Character Seq. No.	No. of Occurrence	% of Occurrence
(218)	10	0.0621	(239)	1	0.0062
(219)	10	0.0621	(240)	24	0.1491
(220)	9	0.0559	(241)	0	0.0000
(221)	1	0.0062	(242)	0	0.0000
(222)	1	0.0062	(243)	0	0.0000
(223)	1	0.0062	(244)	0	0.0000
(224)	1	0.0062	(245)	4	0.0248
(225)	0	0.0000	(246)	5	0.0310
(226)	19	0.1180	(247)	0	0.0000
(227)	1	0.0062	(248)	11	0.0683
(228)	6	0.0372	(249)	0	0.0000
(229)	2	0.0124	(250)	3	0.0186
(230)	1	0.0062	(251)	2	0.0124
(231)	3	0.0186	(252)	0	0.0000
(232)	8	0.0497	(253)	2	0.0124
(233)	1	0.0062	(254)	8	0.0497
(234)	0	0.0000	(255)	2	0.0124
(235)	0	0.0000	(256)	0	0.0000
(236)	1	0.0062	(257)	1	0.0062
(237)	0	0.0000	(258)	0	0.0000
(238)	1	0.0062	(259)	18	0.1118

Table 3.6 Contd.

Character Seq. No.	No. of Occurrence	% of Occurrence	Character Seq. No.	No. of Occurrence	% of Occurrence
(260)	9	0.0559	(281)	1	0.0062
(261)	1	0.0062	(282)	1	0.0062
(262)	1	0.0062	(283)	1	0.0062
(263)	0	0.0000	(284)	0	0.0000
(264)	0	0.0000	(285)	0	0.0000
(265)	0	0.0000	(286)	0	0.0000
(266)	3	0.0186	(287)	0	0.0000
(267)	7	0.0435	(288)	0	0.0000
(268)	0	0.0000	(289)	1	0.0062
(269)	1	0.0062	(290)	1	0.0062
(270)	12	0.0745	(291)	0	0.0000
(271)	20	0.1243	(292)	6	0.0372
(272)	1	0.0062	(293)	0	0.0000
(273)	1	0.0062	(294)	0	0.0000
(274)	0	0.0000	(295)	0	0.0000
(275)	8	0.0497	(296)	0	0.0000
(276)	3	0.0186	(297)	0	0.0000
(277)	9	0.0559	(298)	0	0.0000
(278)	1	0.0062	(299)	0	0.0000
(279)	0	0.0000	(300)	0	0.0000
(280)	0	0.0000	(301)	0	0.0000

Table 3.6 Contd.

Charac- ter Seq. No.	No. of Occurrence	% of Occurrence	Charac- ter Seq. No.	No. of Occurrence	% of Occurrence
(302)	0	0.0000	(322)	8	0.0497
(303)	0	0.0000	(323)	0	0.0000
(304)	0	0.0000	(324)	2	0.0124
(305)	1	0.0062	(325)	0	0.0000
(306)	0	0.0000	(326)	0	0.0000
(307)	1	0.0062	(327)	2	0.0124
(308)	0	0.0000	(328)	0	0.0000
(309)	0	0.0000	(329)	2	0.0124
(310)	0	0.0000	(330)	1	0.0062
(311)	0	0.0000	(331)	0	0.0000
(312)	1	0.0062	(332)	0	0.0000
(313)	0	0.0000	(333)	0	0.0000
(314)	0	0.0000	(334)	2	0.0124
(315)	0	0.0000	(335)	2	0.0124
(316)	0	0.0000	(336)	0	0.0000
(317)	0	0.0000	(337)	1	0.0062
(318)	0	0.0000	(338)	0	0.0000
(319)	0	0.0000	(339)	0	0.0000
(320)	0	0.0000	(340)	0	0.0000
(321)	11	0.0683	(341)	0	0.0000

Table 3.6 Contd.

Character Seq. No.	No. of Occurrence	% of Occurrence	Character Seq. No.	No. of Occurrence	% of Occurrence
(342)	0	0.0000	(363)	0	0.0000
(343)	0	0.0000	(364)	0	0.0000
(344)	0	0.0000	(365)	0	0.0000
(345)	0	0.0000	(366)	0	0.0000
(346)	0	0.0000	(367)	17	0.1056
(347)	0	0.0000	(368)	16	0.0994
(348)	0	0.0000	(369)	6	0.0372
(349)	0	0.0000	(370)	13	0.0807
(350)	0	0.0000	(371)	4	0.0248
(351)	1	0.0062	(372)	4	0.0248
(352)	0	0.0000	(373)	9	0.0559
(353)	0	0.0000	(374)	0	0.0000
(354)	0	0.0000	(375)	1	0.0062
(355)	0	0.0000	(376)	1	0.0062
(356)	0	0.0000	(377)	0	0.0000
(357)	1	0.0062	(378)	0	0.0000
(358)	1	0.0062	(379)	0	0.0000
(359)	0	0.0000	(380)	0	0.0000
(360)	0	0.0000	(381)	0	0.0000
(361)	0	0.0000	(382)	0	0.0000
(362)	0	0.0000	(383)	0	0.0000

Table 3.6 Contd.

Character Seq. No.	No. of Occurrence	% of Occurrence	Character Seq. No.	No. of Occurrence	% of Occurrence
(384)	0	0.0000	(406)	10	0.0621
(385)	0	0.0000	(407)	7	0.0435
(386)	0	0.0000	(408)	54	0.3356
(387)	0	0.0000	(409)	12	0.0745
(388)	1	0.0062	(410)	0	0.0000
(389)	0	0.0000	(411)	0	0.0000
(390)	0	0.0000	(412)	7	0.0435
(391)	0	0.0000	(413)	7	0.0435
(392)	41	0.2548	(414)	0	0.0000
(393)	69	0.4288	(415)	0	0.0000
(394)	47	0.2921	(416)	0	0.0000
(395)	40	0.2486	(417)	0	0.0000
(396)	34	0.2113	(418)	0	0.0000
(397)	47	0.2921	(419)	0	0.0000
(398)	39	0.2423	(420)	0	0.0000
(399)	25	0.1553	(421)	0	0.0000
(400)	37	0.2299	(422)	1	0.0062
(401)	38	0.2361	(423)	0	0.0000
(402)	112	0.6960	(424)	51	0.3169
(403)	3	0.0186	(425)	0	0.0000
(404)	1	0.0062	(426)	0	0.0000
(405)	188	1.1684	(427)	0	0.0000

Table 3.6 Contd.

Character Seq. No.	No. of Occurrence	% of Occurrence
(428)	6	0.0372
(429)	6	0.0372
(430)	0	0.0000
(431)	0	0.0000
(432)	0	0.0000
(433)	0	0.0000
(434)	0	0.0000
SPACE	2,927	18.1914

occurrence, which is given in Table 3.7.

'Space' has not been counted as a character in the survey made by Prabir Kumar Das (Table B-1). For computing the frequency of occurrence of Bengali Graphic Symbols, no. of occurrence of 'Space' for this survey has been estimated as discussed below.

It has been assumed that the observations in all these three surveys are random [9], i.e.,

- the method of sampling was unbiased and all samples were collected in an idealistic condition,
- though there is linguistic relationship among the characters being observed, for the sake of simplicity it has been assumed that there is no relation among the occurrences of the characters, i.e., each character in the universe of the printed Bengali text has the same chance of occurring.

For the case of random sampling, this type of variate, i.e., the no. of occurrence of a character in a large volume of printed text follows Poisson distribution[10]. The estimation of Poisson probability is expressed by the equation [9]

$$P(c) = e^{-m} \cdot \frac{m^c}{c!} \qquad (3.1)$$

where, $e = 2.71828$

$m = Np$

N = size of sample

p = basic probability

c = no. of items the event in consideration is occurred.

As all three sets of data of Table 3.7 have been observed in the same universe of printed Bengali text, the Poisson probability of any fixed no. of occurrence of 'Space' in a fixed size of sample will be equal for all three cases. Under this assumption, the Poisson probability of expected no. of occurrence of 'Space' being estimated for set 2 can be estimated as the average of the Poisson probabilities of actual occurrence of 'Space' for set 1 and set 3 for the same size of sample for all three sets. From this estimated no. of occurrence of 'Space' for the assumed size of sample, the no. of occurrence of 'Space' for set 2 can be back calculated.

As the computation of factorial is involved in the estimation of Poisson probability, the assumed size of the sample is so chosen that the corresponding no. of occurrence of 'Space' for each set of data lies within the computational limit of the existing machines. For the present cases, it has been found that a sample size of 400 satisfies the above requirement and for this purpose all data of all the sets have been transferred to the scale of 400.

For set 1, Poisson probability of observed no. of occurrence of 'Space' in the scale of 400 is

$$P(c_1) = e^{-m_1} \cdot \frac{m_1^{c_1}}{c_1!}$$

$$= e^{(-\frac{400}{132})} \cdot \frac{(\frac{400}{132})^{66}}{66!} = 5.32306 \times 10^{-63}$$

where, $m_1 = N_1\ p_1 = 400.\ \frac{1}{132}$

N_1 = sample size = 400

p_1 = basic probability = $\frac{1}{132}$

c_1 = no. of occurrence of 'Space' in the scale of 400 = 66.1837 i.e., 66.

For set 3, Poisson probability of observed no. of occurrence of 'Space' in the scale of 400 is

$$p\ (c_3) = e^{-m_3} \cdot \frac{m_3^{c_3}}{c_3!}$$

$$= e^{(-\frac{400}{132})} \cdot \frac{(\frac{400}{132})^{67}}{67!} = 2.40753 \times 10^{-64}$$

where, $m_3 = N_3 p_3 = 400 \cdot \frac{1}{132}$

N_3 = sample size = 400

p_3 = basic probability $= \frac{1}{132}$

c_3 = no. of occurrence of 'Space' in the scale of 400 = 66.9411 i.e., 67.

For set 2, Poisson probability of expected no. of occurrence of 'Space' being estimated in the scale of 400 is

$$p\,(c_2) = \frac{p(c_1) + p(c_3)}{2} = e^{-m_2} \cdot \frac{m_2^{\,c_2}}{c_2!}$$

$$\text{or, } \frac{5.32306 \times 10^{-63} + 2.40753 \times 10^{-64}}{2} = e^{(-\frac{400}{132})} \cdot \frac{(\frac{400}{132})^{c_2}}{c_2!}$$

$$\text{or, } 5.75952 \times 10^{-62} \;\; c_2! = (3.030303)^{c_2}$$

$$\therefore \; f(c_2) = 5.75952 \times 10^{-62} \;\; c_2! - (3.030303)^{c_2} = 0 \qquad (3.2)$$

where, $m_2 = N_2 p_2 = 400 \cdot \frac{1}{132}$

N_2 = sample size = 400

p_2 = basic probability $= \frac{1}{132}$

c_2 = no. of expected occurrence of 'Space' being estimated in the scale of 400.

The equation (3.2) is solved by using Incremental-Search Method[11] and the incremental value of c_2 is taken an integer as factorial of a fractional number can not be computed. Between two consecutive values of c_2 with unit increment, where the function $f(c_2)$ changes its sign, that value of c_2 is taken as the final value of c_2 whose absolute value is smaller, i.e., which approaches more towards the zero. Here

$$f(66) = 5.75952 \times 10^{-62} \times 66! - (3.030303)^{66} = -2.8638 \times 10^{31}$$

$$\text{and } f(67) = 5.75952 \times 10^{-62} \times 67! - (3.030303)^{67} = 1.91877 \times 10^{33}$$

The function $f(c_2)$ changes its sign between the value of c_2 of 66 and 67 and as the absolute value of f(66) is smaller than the absolute value of f(67), the final value of c_2 is taken as 66 and no other approximation using other method is done because c_2 can not be a fractional number as its factorial is involved in $f(c_2)$.

The estimated no. of occurrence of 'Space' for set 2 in the scale of 400 can be expressed as

$$C_2 = \frac{X_{sp}}{\sum_{i=1}^{131} X_i + X_{sp}} \times 400 \qquad (3.3)$$

where, X_{sp} = estimated no. of occurrence of 'Space' for set 2

X_i = no. of occurrence of ith BGS for set 2.

From equation (3.3), the estimated no. of occurrence of 'Space' for set 2 is

$$X_{sp} = \frac{C_2 \sum_{i=1}^{131} X_i}{400 - C_2} = \frac{66 \times 45809}{400 - 66}$$

$$= 9052.0778$$

$$\text{i.e., } 9052.$$

The frequency distribution of occurrence of the Bengali Graphic Symbol Set is shown in Fig. 3.4 . The distribution is not uniform-it ranges from 0.0000% to a peak frequency of 16.5515% for 'Space'.

3.7 KEY-BOARD LAY-OUT OF THE BENGALI GRAPHIC PRIMITIVES

As the frequency distribution of occurrence of the Bengali Graphic Symbols (Fig. 3.4) is not uniform, no lexical ordering is possible to be maintained, for ensuring the enhancement of typing speed, in devicing the key-board lay-out of the Bengali Graphic Primitives. On the other hand, the nos. of the Bengali Graphic Symbols are 131 which is much to be accomodated on a handy size of key-board with maintaining lexical similarities and ordering. However, two key-board lay-outs have been proposed-one with 56 main keys and other with 47 main keys similar to a QWERTY style English key-board, i.e., a typical key-board that begins with these six letters, left -to-right, in the top row below the numerals.

Table 3.7: Frequency of Occurrence (on the basis of 140,688 no. of occurrence) of Bengali Graphic Symbols (BGS)

BGS Seq. No.	Set 1 from Table A-2		Set 2 from Table B-1		Set 3 from Table 3.6		Total Set 1 + Set 2+ set 3	
	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	% of occurrence
(01)	1,121	6.5616	939	6.8464	224	5.1229	2,284	1.6234
(02)	754	4.4134	691	5.0382	160	3.6592	1,605	1.1408
(03)	700	4.0973	41	0.2989	0	0.0000	741	0.5267
(04)	233	1.3638	263	1.9176	55	1.2579	551	0.3916
(05)	12	0.0702	0	0.0000	0	0.0000	12	0.0085
(06)	3	0.0176	2	0.0146	0	0.0000	5	0.0036
(07)	634	3.7110	555	4.0466	187	4.2767	1,376	0.9781
(08)	20	0.1171	20	0.1458	3	0.0686	43	0.0306
(09)	393	2.3004	340	2.4790	110	2.5157	843	0.5992
(10)	2	0.0117	0	0.0000	1	0.0229	3	0.0021
(11)	253	1.4809	0	0.0000	62	1.4180	315	0.2239
(12)	26	0.1522	48	0.3500	12	0.2744	86	0.0611
(13)	2,522	14.7621	2,260	16.4780	546	12.4871	5,328	3.7871
(14)	380	2.2243	342	2.4936	84	1.9211	806	0.5729
(15)	726	4.2495	411	2.9967	197	4.5054	1,334	0.9482
(16)	120	0.7024	71	0.5177	29	0.6632	220	0.1564
(17)	13	0.0761	30	0.2187	5	0.1144	48	0.0341
(18)	587	3.4359	297	2.1655	120	2.7444	1,004	0.7136
(19)	567	3.3188	356	2.5956	180	4.1165	1,103	0.7840
(20)	654	3.8281	519	3.7841	165	3.7736	1,338	0.9510

Table 3.7 Contd.

BGS Seq. No.	Set 1 from Table A-2		Set 2 from Table B-1		Set 3 from Table 3.6		Total Set 1 + Set 2 + Set 3	
	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	% of occurrence
(21)	53	0.3102	33	0.2406	16	0.3659	102	0.0725
(22)	6	0.0351	0	0.0000	0	0.0000	6	0.0043
(23)	633	3.7052	488	3.5581	197	4.5054	1,318	0.9368
(24)	146	0.8546	131	0.9551	27	0.6175	304	0.2161
(25)	98	0.5736	65	0.4739	54	1.2350	217	0.1542
(26)	20	0.1171	29	0.2114	3	0.0686	52	0.0370
(27)	228	1.3346	326	2.3769	69	1.5780	623	0.4428
(28)	2,441	14.2880	2,189	15.9603	519	11.8696	5,149	3.6599
(29)	380	2.2243	386	2.8144	123	2.8130	889	0.6319
(30)	1,160	6.7899	936	6.8245	308	7.0440	2,404	1.7087
(31)	493	2.8857	282	2.0561	115	2.6301	890	0.6326
(32)	2,642	15.4645	1,871	13.6417	659	15.0715	5,172	3.6762
(33)	1,260	7.3752	1,114	8.1223	370	8.4620	2,744	1.9504
(34)	117	0.6848	68	0.4958	43	0.9834	228	0.1621
(35)	2,242	13.1232	1,646	12.0012	513	11.7324	4,401	3.1282
(36)	426	2.4935	271	1.9759	103	2.3556	800	0.5686
(37)	1,460	8.5459	1,169	8.5234	374	8.5535	3,003	2.1345
(38)	400	2.3413	433	3.1571	83	1.8982	916	0.6511
(39)	3,698	21.6457	3,002	21.8880	875	20.0114	7,575	5.3843
(40)	1,655	9.6873	1,041	7.5901	376	8.5992	3,072	2.1836
(41)	754	4.4134	495	3.6091	162	3.7050	1,411	1.0029
(42)	206	1.2058	164	1.1957	34	0.7776	404	0.2872
(43)	1,545	9.0434	944	6.8828	406	9.2853	2,895	2.0577

Table 3.7 Contd.

BGS Seq. No.	Set 1 from Table A-2		Set 2 from Table B-1		Set 3 from Table 3.6		Total Set 1 + Set 2 + Set 3	
	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	% of occurrence
(44)	809	4.7354	840	6.1246	195	4.4597	1,844	1.3107
(45)	1,214	7.1060	939	6.8464	249	5.6947	2,402	1.7073
(46)	293	1.7150	214	1.5603	62	1.4180	569	0.4044
(47)	0	0.0000	3	0.0219	1	0.0229	4	0.0028
(48)	73	0.4273	110	0.8020	41	0.9377	224	0.1592
(49)	7,122	41.6875	6,210	45.2780	1,691	38.6735	15,023	10.6782
(50)	3,277	19.1814	2,641	19.2559	732	16.7410	6,650	4.7268
(51)	693	4.0564	471	3.4341	133	3.0417	1,297	0.9219
(52)	1,137	6.6553	617	4.4986	279	6.3808	2,033	1.4450
(53)	175	1.0243	148	1.0791	49	1.1206	372	0.2644
(54)	146	0.8546	154	1.1228	40	0.9148	340	0.2417
(55)	5,780	33.8323	4,406	32.1248	1,501	34.3282	11,687	8.3070
(56)	46	0.2693	39	0.2844	20	0.4574	105	0.0746
(57)	46	0.2693	58	0.4229	6	0.1372	110	0.0782
(58)	112	0.6556	104	0.7583	23	0.5360	239	0.1699
(59)	3	0.0176	2	0.0146	2	0.0457	7	0.0050
(60)	126	0.7375	1	0.0073	40	0.9148	167	0.1187
(61)	53	0.3104	31	0.2260	22	0.5031	106	0.0753
(62)	6	0.0351	4	0.0292	4	0.0915	14	0.0100
(63)	33	0.1932	37	0.2698	42	0.9605	112	0.0796
(64)	4	0.0234	0	0.0000	1	0.0229	5	0.0036
(65)	42	0.2458	6	0.0437	11	0.2516	59	0.0419
(66)	80	0.4683	29	0.2114	24	0.5289	133	0.0945

Table 3.7 Contd.

DGS Seq. No.	Set 1 from Table A-2		Set 2 from Table B-1		Set 3 from Table 3.6		Total Set 1 + Set 2 + Set 3	
	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	% of occurrence
(67)	480	2.8096	202	1.4728	123	2.8130	805	0.5722
(68)	26	0.1522	7	0.0510	7	0.1601	40	0.0284
(69)	13	0.0761	9	0.0656	3	0.0686	25	0.0178
(70)	66	0.3863	76	0.5541	39	0.8919	181	0.1287
(71)	13	0.0761	41	0.2989	32	0.7318	86	0.0611
(72)	33	0.1932	30	0.2187	19	0.4345	82	0.0583
(73)	106	0.6205	113	0.8239	31	0.7090	250	0.1777
(74)	206	1.2058	192	1.3999	58	1.3265	456	0.3241
(75)	325	1.9023	283	2.0634	110	2.5157	718	0.5103
(76)	2	0.0117	0	0.0000	0	0.0000	2	0.0014
(77)	26	0.1522	12	0.0875	3	0.0686	41	0.0291
(78)	246	1.4399	34	0.2479	15	0.3431	295	0.2097
(79)	86	0.5034	27	0.1969	4	0.0915	117	0.0832
(80)	593	3.4710	531	3.8716	180	4.1166	1,304	0.9269
(81)	689	4.0330	496	3.6164	203	4.6427	1,388	0.9866
(82)	40	0.2341	6	0.0437	1	0.0229	47	0.0334
(83)	120	0.7024	137	0.9989	52	1.1893	309	0.2196
(84)	26	0.1522	29	0.2114	3	0.0686	58	0.0412
(85)	4	0.0234	0	0.0000	1	0.0229	5	0.0036
(86)	80	0.4683	58	0.4229	15	0.3431	153	0.1088
(87)	0	0.0000	0	0.0000	0	0.0000	0	0.0000
(88)	2	0.0117	8	0.0583	1	0.0229	11	0.0078
(89)	0	0.0000	5	0.0365	1	0.0229	6	0.0043
(90)	0	0.0000	0	0.0000	10	0.2287	10	0.0071

Table 3.7 Contd.

BGS Seq. No.	Set 1 from Table A-2		Set 2 From Table B-1		Set 3 from Table 3.6		Total Set 1 + Set 2 + Set 3	
	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	% of occurrence
(91)	73	0.4273	64	0.4666	1	0.0229	138	0.0981
(92)	186	1.0887	76	0.5541	32	0.7318	294	0.2090
(93)	4	0.0234	0	0.0000	41	0.9377	45	0.0320
(94)	14	0.0819	0	0.0000	69	1.5780	83	0.0590
(95)	1	0.0059	0	0.0000	47	1.0749	48	0.0341
(96)	1	0.0059	0	0.0000	40	0.9148	41	0.0291
(97)	2	0.0117	0	0.0000	34	0.7776	36	0.0256
(98)	2	0.0117	0	0.0000	47	1.0749	49	0.0348
(99)	1	0.0059	0	0.0000	39	0.8919	40	0.0284
(100)	4	0.0234	0	0.0000	25	0.5718	29	0.0206
(101)	0	0.0000	0	0.0000	37	0.8462	37	0.0263
(102)	0	0.0000	0	0.0000	38	0.8691	38	0.0270
(103)	76	0.4449	199	1.4509	112	2.5615	387	0.2751
(104)	4	0.0234	213	1.5530	3	0.0686	220	0.1564
(105)	0	0.0000	74	0.5395	1	0.0229	75	0.0533
(106)	92	0.5385	513	3.7404	188	4.2996	793	0.5637
(107)	9	0.0527	22	0.1604	10	0.2287	41	0.0291
(108)	1	0.0059	11	0.0802	7	0.1601	19	0.0135
(109)	140	0.8195	775	5.6506	66	1.5094	981	0.6973
(110)	15	0.0878	56	0.4083	0	0.0000	71	0.0505
(111)	0	0.0000	178	1.2978	14	0.3202	192	0.1365
(112)	0	0.0000	0	0.0000	0	0.0000	0	0.0000

Table 3.7 Contd.

DGS Seq. No.	Set 1 from Table A-2		Set 2 from Table B-1		Set 3 from Table 3.6		Total Set 1+ Set 2 + Set 3	
	No. of occurrence	No. of occurrence in 400 scale	No. of occurrence	No. of occurrence in 400 scale	No. of ocurrence	No. of occurrence in 400 scale	No. of occurrence	% of occurrence
(113)	0	0.0000	0	0.0000	0	0.0000	0	0.0000
(114)	0	0.0000	0	0.0000	0	0.0000	0	0.0000
(115)	0	0.0000	0	0.0000	0	0.0000	0	0.0000
(116)	0	0.0000	0	0.0000	0	0.0000	0	0.0000
(117)	0	0.0000	0	0.0000	0	0.0000	0	0.0000
(118)	0	0.0000	0	0.0000	0	0.0000	0	0.0000
(119)	0	0.0000	0	0.0000	1	0.0229	1	0.0007
(120)	0	0.0000	0	0.0000	0	0.0000	0	0.0000
(121)	0	0.0000	0	0.0000	51	1.1664	51	0.0363
(122)	170	0.9951	0	0.0000	0	0.0000	170	0.1208
(123)	0	0.0000	0	0.0000	0	0.0000	0	0.0000
(124)	0	0.0000	0	0.0000	0	0.0000	0	0.0000
(125)	0	0.0000	0	0.0000	6	0.1372	6	0.0043
(126)	0	0.0000	0	0.0000	6	0.1372	6	0.0043
(127)	0	0.0000	0	0.0000	0	0.0000	0	0.0000
(128)	0	0.0000	0	0.0000	0	0.0000	0	0.0000
(129)	0	0.0000	0	0.0000	0	0.0000	0	0.0000
(130)	0	0.0000	0	0.0000	0	0.0000	0	0.0000
(131)	0	0.0000	0	0.0000	0	0.0000	0	0.0000
SPACE	11,307	66.1837	9,052	65.9995	2,927	66.9411	23,286	16.5515

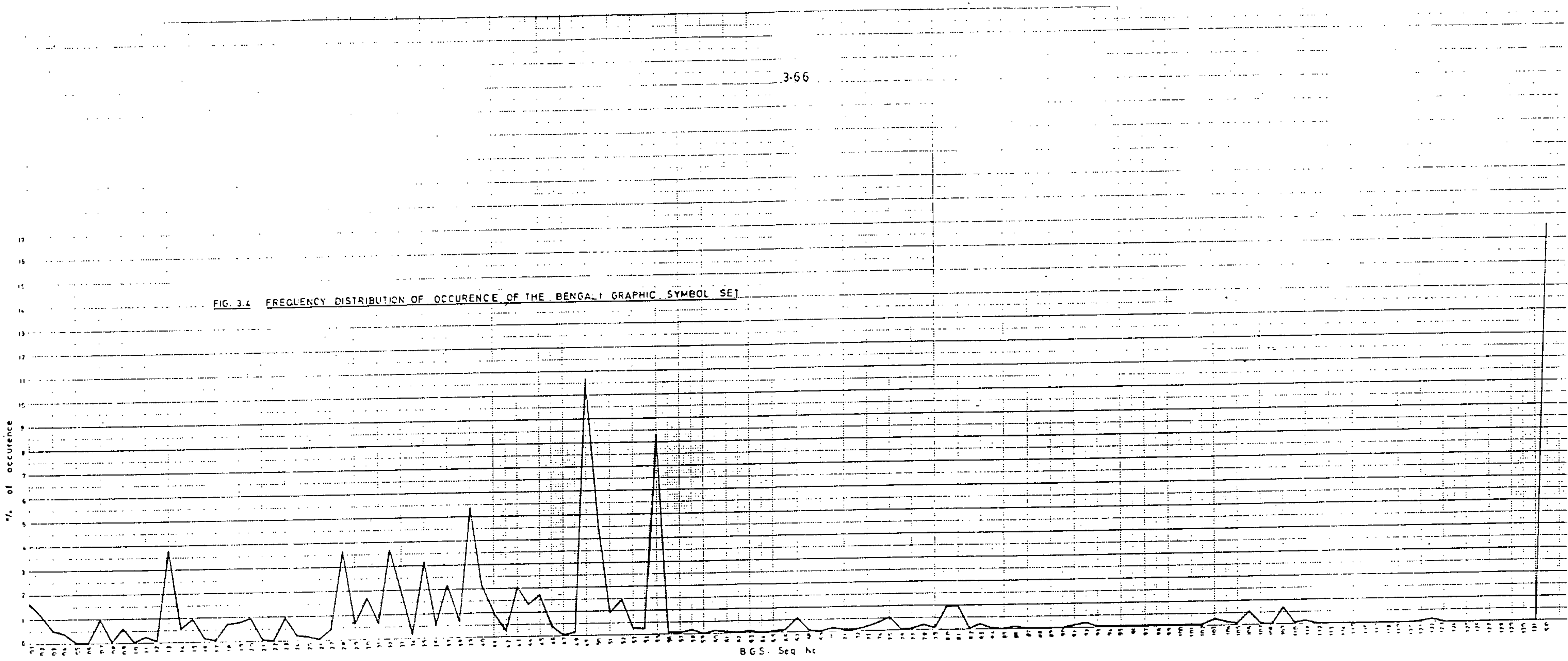


FIG. 3.4 FREQUENCY DISTRIBUTION OF OCCURENCE OF THE BENGALI GRAPHIC SYMBOL SET

To make proper load distribution on all the active figures and to enhance the typing speed, key-board lay-outs have been deviced under the assumptions that

- Two shifts are used to assign three symbols to some keys (one normal and two shift symbols) and two symbols to other keys (one normal and one shift symbol) for accomodating all the 131 Bengali Graphic Symbols on a handy size of key-board.

- Common symbols of English and Bengali are kept, as much as possible, at the same position of English key-board for facilitating typing English and Bengali by the same typist.
- Main lexical symbols are kept in the bottom three lines with keeping special symbols in the top line for ensuring easy access to lexical symbols.
- Most frequent symbols are kept at the middle of the middle two lines and other less frequent symbols are kept at the two sides of the middle two lines and at the bottom line for enhancing the typing speed.
- Main symbol, special shaped symbol and fala symbol of a varna are kept at the same key for ensuring easy refference to the symbols.
- In some cases, different symbols but having closest relations are kept on the same key for accomodating all symbols on the limited keys.

- Symbols of swara varnas and their kar symbols are kept at the same key for ensuring easy refference to them.
- Swara-kars are kept at the normal position with keeping Swara Varnas at the shift 1 position, as the swara-kars are much more frequent than the Swara Varnas.
- Main symbol, among symbols assigned to a key, is kept at the normal position with keeping special-shaped symbol at the shift 1 position and fala or other closely related symbol at the shift 2 position for ensuring easy refference to the symbols.
- In most cases, unless otherwise guided by other assumptions, most frequent symbol among symbols assigned to a key is kept at the normal position and other symbol or symbols are kept at the shift position for reducing the average key-stroke per symbol.

Key-board lay-out with 56 main keys has been deviced with the aim to design a new Bengali key-board of handy size with other assumptions mentioned earlier. This key- board lay-out is shown in Fig. 3.5 and named the BC II key-board, i.e., key-board for producing BCII codes. The other key-board lay-out with 47 main keys has been deviced with the aim to change the existing QWERTY Style English key-board to a Bengali key-board. This key-board lay-out is shown in Fig. 3.6 and named the ABCII key-board, i.e., adapted BC II key-board. The first one is suggested for final

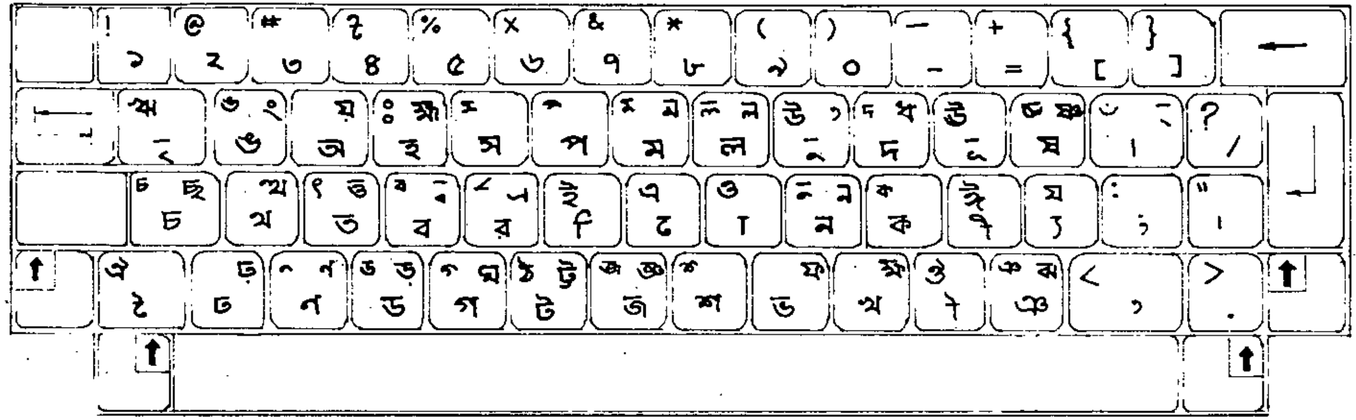


FIG. 3.5 KEY BOARD LAY OUT OF THE BENGALI GRAPHIC PRIMITIVES WITH 56 MAIN KEYS (BCII KEY BOARD)

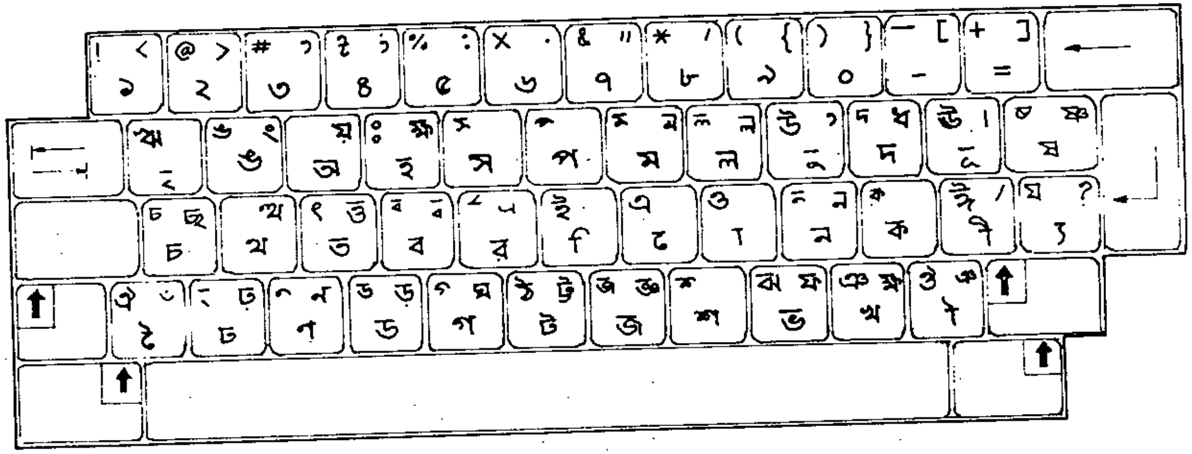


FIG 3.6 KEY BOARD LAY-OUT OF THE BENGALI GRAPHIC PRIMITIVES WITH 47 MAIN KEYS (ABCII KEY BOARD)

use and the second one is suggested for present use until the first one is introduced completely.

3.8 STATISTICAL CONSIDERATIONS IN DEVICING KEY-BOARD LAY-OUTS OF THE BENGALI GRAPHIC PRIMITIVES

To enhance the typing speed, one should attempt to reduce the average key-stroke per symbol required, i.e., to reduce the no. of symbols where shift-key-stroke is required. As the shift keys can not be eliminated, it has been attempted that the Graphic Symbols which occur most frequently should be placed at the normal key-positions under other assumptions mentioned earlier. In order to achieve this, the frequency of occurrence of Bengali Graphic Symbols of Table 3.7 has been used. The summary of this statistics is given in Table 3.8.

Several factors are worth noting in Table 3.8 : -

- The frequency distribution is not uniform.
- 32 graphic symbols (with individual frequencies of 0.6% and above: Called Group I) account for 72.3727% of the total occurrence.
- Another 32 graphic symbols (with inidividual frequencies of 0.1% and above but below 0.6%: Called Group II) take up 9.3726% of the total occurrence.

- Rest 67 graphic symbols (with individual frequencies of below 0.1% : Called Group III) take up the remaining 1.7030% of the total occurrence.

In the key-board lay-out with 56 main keys (BC II key-board), from Group I, 25 symbols with cummulative frequency of 65.4922% have been allocated to normal key-position, 3 symbols with cummulative frequency of 2.7700% have been allocated to shift 1-key-position and remaining 4 symbols with cummulative frequency of 4.1105% have been allocated to shift 2- key-position. From Group II, 12 symbols with cummulative frequency of 3.7843% have been allocated to normal key-position, 13 symbols with cummulative frequency of 4.1034% have been allocated to shift 1-key-position and remaining 7 symbols with cummulative frequency of 1.4849% have been allocated to shift 2- key position. From Group III, 19 symbols with cummulative frequency of 0.5814% have been allocated to normal key-position, 35 symbols with cummulative frequency of 0.7371% have been allocated to shift 1- key-position and remaining 13 symbols with cummulative frequency of 0.3845% have been allocated to shift 2- key-position. This results to allocation of 56 symbols with cummulative frequency of 69.8579% to normal key-position, 51 symbols with cummulative frequency of 7.6105% to shift 1-key-position and remaining 24 symbols with cummulative frequency of 5.9799% to shift 2- key-position, leading to

Average Character/Key,

$$C_{avl} = \frac{N_c}{N_{kl}} = \frac{131}{56} = 2.3393$$

where, C_{avl} = average character/key in BCII key-board

N_c = total nos. of characters

N_{kl} = total nos. of keys in BCII key-board

and

Average Key-Stroke/Character,

$$S_{avl} = \frac{\sum_{i=1}^{2} i.c_i}{\sum_{i=1}^{2} c_i} = \frac{1 \times 86.4094 + 2 \times 13.5904}{86.4094 + 13.5904}$$

$$= 1.1359$$

where, S_{avl} = average key-stroke/character in BC II key-board

C_i = % cummulative frequency for i-stroke symbols

i = 1 for single-stroke symbols & space

= 2 for double-stroke symbols.

In the key-board lay-out with 47 main keys (ABC II key-board), from Group I, 25 symbols with cummulative frequency of 65.4922% have been allocated to normal key-position, 3 symbols with cummulative frequency of 2.7700% have been allocated to shift 1-key-position and remaining 4 symbols with cummulative frequency of 4.1105% have been allocated to shift 2- key-position. From Group II, 8 symbols with cummulative frequency of 2.6526% have been allocated to normal key-position, 12 symbols with cummulative frequency of

3.8944% have been allocated to shift 1-key-position and remaining 12 symbols with cummulative frequency of 2.8256% have been allocated to shift 2-key-position. From Group III , 14 symbols with cummulative frequency of 0.5408% have been allocated to normal key-position, 30 symbols with cummulative frequency of 0.6995% have been allocated to shift 1- key-position and remaining 23 symbols with cummulative frequency of 0.4627% have been allocated to shift 2-key-position. This results to allocation of 47 symbols with cummulative frequency of 68.6856% to normal key- position, 45 symbols with cummulative frequency of 7.3639% to shift 1-key-position and remaining 39 symbols with cummulative frequency of 7.3988% to shift 2 -key-position, leading to

Average Character/Key,

$$C_{av2} = \frac{N_c}{N_{k2}} = \frac{131}{47} = 2.7872$$

where, C_{av2} = average character/key in ABC II key-board

N_c = total nos. of characters

N_{k2} = total nos. of keys in ABC II key-board

and

Average Key-Stroke/Character,

$$S_{av2} = \frac{\sum_{i=1}^{2} i.c_i}{\sum_{i=1}^{2} c_i} = \frac{1 \times 85.2371 + 2 \times 14.7627}{85.2371 + 14.7627}$$

$$= 1.1476$$

where, S_{av2} = average key-stroke/character in ABCII key-board

C_i = % cummulative frequency for i-stroke symbols

i = 1 for single-stroke symbols & space

= 2 for double -stroke symbols.

3.9 COMPARATIVE STUDY OF THE PROPOSED KEY-BOARDS WITH THE OPTIMA MUNIR KEY-BOARD

The only available Bengali key-board used in Bangladesh is the 'Optima Munir' key-board which has the following inherent drawbacks which makes it unsuitable for computer applications:

i) It has some symbols, used for generating Bengali Varnas by superposition or concatenation with other symbols, which do not have any lexical identify.If this key-board is used in computer applications, lexical analysis will not be possible with these symbols.

ii) It requires superposition or concatenation of more than one symbols for generating some frequently used Bengali Varnas which is difficult and cumbersome for a typist. Moreover, this will make the computer slow because the superposition is to be effected either by software or by inbuilt hardware decision logic.

Table 3.8 : Summary Statistics of Bengali Graphic Symbols (BGS)

Group (on the basis of % of occurrence)	Total No. of BGS	Cummulative frequency in %	Position in key-boards											
			BCII key-board						ABCII key-board					
			Normal position		Shift 1 position		Shift 2 position		Normal position		Shift 1 position		Shift 2 position	
			No. of BGS	Cummulative frq. in %	No. of BGS	Cummulative frq. in %	No. of BGS	Cummulative frq. in %	No. of BGS	Cummulative frq. in %	No. of BGS	Cummulative Frq. in %	No. of BGS	Cummulative frq. in %
I (0.6% & above)	32	72.3727	25	65.4922	3	2.7700	4	4.1105	25	65.4922	3	2.7700	4	4.1105
II (0.1% & above but below 0.6%)	32	9.3726	12	3.7843	13	4.1034	7	1.4849	8	2.6526	12	3.8944	12	2.8256
III (Below 0.1%)	67	1.7030	19	0.5814	35	0.7371	13	0.3845	14	0.5408	30	0.6995	23	0.4627
Total	131	83.4483	56	69.8579	51	7.6105	24	5.9799	47	68.6856	45	7.3639	39	7.3988
SPACE		16.5515												

iii) It generates most of the compound byanjana varnas by concatenation or by superposition of the constituent varnas of normal shape and size, which highly hampers the legibility of the compound byanjana varnas and for that, the compound byanjana varnas can not be generated in their original shapes.

iv) It does not have some frequently used special graphic symbols which are essential in computer applications.

On the other hand, this sorts of drawbacks have been removed from the proposed key-boards. Each of the 131 Bengali Graphic Symbols accomodated on the proposed key-boards has its own unique lexical identity and no superposition is required to generate these graphic symbols. All of the compound byanjana varnas, except that with ্র-fala, are generated by concatenation of specially shaped varnas, normal shaped varnas and fala symbols as required by the compound byanjana varnas such that the shape of the compound byanjana varnas correspond to the shape presently used in lino-type. Most of the widely used special graphic symbols have been accomodated in these proposed key-boards.

The 'Optima Munir' key-board also has the following problems with its key-board lay-out (appendix-C):

i) The ও ড গ ঙ ঞ varnas have been placed in the top line. As the frequency of occurrence of these varnas are considerably high, placement of these varnas reduces the typing speed. Instead of these varnas, least frequent special graphic symbols could be placed in the top line and these varnas could be placed in the middle two lines for increasing the typing speed.

ii) The উ ঋ ঈ varnas and the = / x symbols have been placed in the middle two lines, though their frequency of occurrence is considerably low. These varnas and symbols could be placed in the top line or at the extreme right or left of the middle two lines.

iii) The numerals have been placed at the upper case of the top line in the order of ১২৩৪৫০৯৮৭৬. This ordering makes logical problem with numerical ordering. The numerals could be placed in the order of ১২৩৪৫৬৭৮৯০.

On the other hand, the proposed key-board lay-outs have been deviced based on the frequency of occurrence of the graphic symbols and most frequent symbols have been placed in the middle of the middle two lines and other less frequent symbols have been placed in the top line, bottom line and at the two sides of the middle two lines. Special graphic symbols have been placed in the top line and numerals are ordered in the order of ১২৩ ৪ ৫৬ ৭৮ ৯ ০ . Normal symbol, specially shaped symbol and fala symbol of a varna have been placed in a single key and other symbols have been placed with logical ordering such that the position of the graphic symbols can be remembered easily.

The 'Optima Munir' key-board accomodates only 92 symbols, on the other hand, the proposed key-boards accomodate 131 Bengali Graphic Symbols. Moreover,17 graphic symbols of Table 3.4 requires two impressions to be generated by 'Optima Munir' key-board, among which some are of considerably high frequency of occurrence. Beside these, 14 graphic symbols of Table 3.4 can not be generated by the 'Optima Munir' key-board, among which some are lexical symbols with considerable frequency of occurrence. List of these symbols along with their position based on the frequency of occurrence is given in Table 3.9.

Using the frequency of occurrence of Bengali Graphic Symbols of Table 3.7, it has been calculated that the cumulative frequency of the single stroke symbols of 'Optima Munir' key board and the space is 93.7353 and that of the two-stroke symbols, i.e., which requires an addition shift-key stroke,

Table 3.9: List of Bengali Graphic Symbols Which Require Two Impressions To Be Generated And That Which Can Not Be Generated By Optima Munir Key-Board

Position Based on Freq. of Occr.	Symbol	Superposition or Concatenation of Two Symbols Needed	Can Not Be Generated
14	য়	x	
18	ই	x	
24	ট	x	
41	ড়	x	
42	উ	x	
64	ত্ত	x	
68	ু	x	
69	ি	x	
71	ৈ	x	
77	ঃ		x
78	″	x	
79	ক্ত	x	
88	ঐ	x	
99	!	x	
102	ক্ষ	x	
104	ঞ্চ	x	
106	ক্ষ্ম		x
111	ট্ট		x
112	ঢ়	x	
114	ঔ	x	

Table 3.9 Contd.

Position Based on Freq. of Occr.	Symbol	Superposition or Concatenation of Two Symbos Needed	Can not be Generated
116	ચ		x
117	#		x
121	<		x
122	>		x
125	&		x
126	@		x
127	{		x
128	}		x
129	[		x
130	]		x
131	—		x

is 11.2784. This leads to

Average key-stroke/symbol,

$$K_{av}^{m} = \frac{\sum_{i=1}^{2} i.\ c_i^m}{\sum_{i=1}^{2} C_i^m} = \frac{1 \times 93.7353 + 2 \times 11.2784}{93.7353 + 11.2784}$$

$$= 1.1074$$

where K_{av}^{m} = average key-stroke/ symbol for Optima Munir key-board

C_i^m = cumulative frequency of i-th stroke symbol for Optima Munir key-board

i = 1 for single stroke symbols & space

= 2 for double stroke symbols.

On the other hand, the average key-stroke/character for the BCII key-board is 1.1359 and that for the ABCII key-board is 1.1476. From the above data, it is obvious that the average key-stroke/character for Optima Munir key-board and that for the proposed key-boards are more or less equal. Moreover, the proposed key-boards over come all of the inherent drawbacks of the Optima Munir key-board and provides opportunity for generating much more legible varnas with a logically organized key-board lay-out.

CHAPTER - 4

REPRESENTING BENGALI TEXT IN VARIOUS SOFT-COPY AND HARD-COPY PRINTING DEVICES

4.1 INTRODUCTION

An important objective of text processing is to have a visual feedback of the text entered through the key-board on a Video Display Unit (VDU) and to have a soft-copy or a hard-copy printing of the processed text on a VDU or a hard-copy printer. As the 131 Bengali Graphic Symbols of Table 3.4 have been selected for generating all possible Bengali graphic molecules and the same graphic symbols have been selected as key-board primitives for entering text into a computer, representing these 131 graphic symbols in various soft-copy and hard-copy printing devices is required. Guide line to such representation schemes is discussed in this chapter.

4.2 REPRESENTING BENGALI TEXT IN VIDEO DISPLAY UNIT

The 131 Bengali Graphic Symbol set (BGS) can be represented in VDU in two ways:

- in graphic mode, and
- in character mode.

But these Bengali graphic symbols are of unequal horizontal pitch, i.e., they are of different widths. Usually these symbols are of 0.5 unit, 1.0 unit and 1.5 unit widths. These variable width requirement can easily be satisfied by using graphic mode CRT controller. But the typically used character mode raster-scan VDUs can display only 96 fixed width characters. By sacrificing the variable width requirement, these 131 graphic symbol set can be represented in typically available 96 character fixed width character mode VDU by carefully selecting a 96 character set by which the 131 graphic symbols can be mapped. These 131 variable width graphic symbols can, however, be represented by satisfying variable width requirement in the specially designed variable width character mode raster-scan VDU[3,7] by spliting up the variable width symbols into a set of fixed width sub-symbols.

4.3 REPRESENTING BENGALI TEXT IN DOT MATRIX PRINTER

Dot matrix printers are widely used for hard-copy generation in those text processing systems where solid font letters of the target language are not available. Bengali text can, easily, be represented in the dot matrix printers and this can be done in two ways:

- in graphic mode, and
- in character mode.

In graphic mode, all the 131 Bengali graphic symbols can be represented by satisfying the variable width requirements.
A variable width dot matrix printer controller can, also, be designed for representing all the 131 Bengali graphic symbols

for satifying the variable width requirement by spliting up the variable width symbols into a set of fixed width sub-symbols.

4.4 REPRESENTING BENGALI TEXT IN SOLID FONT PRINTER

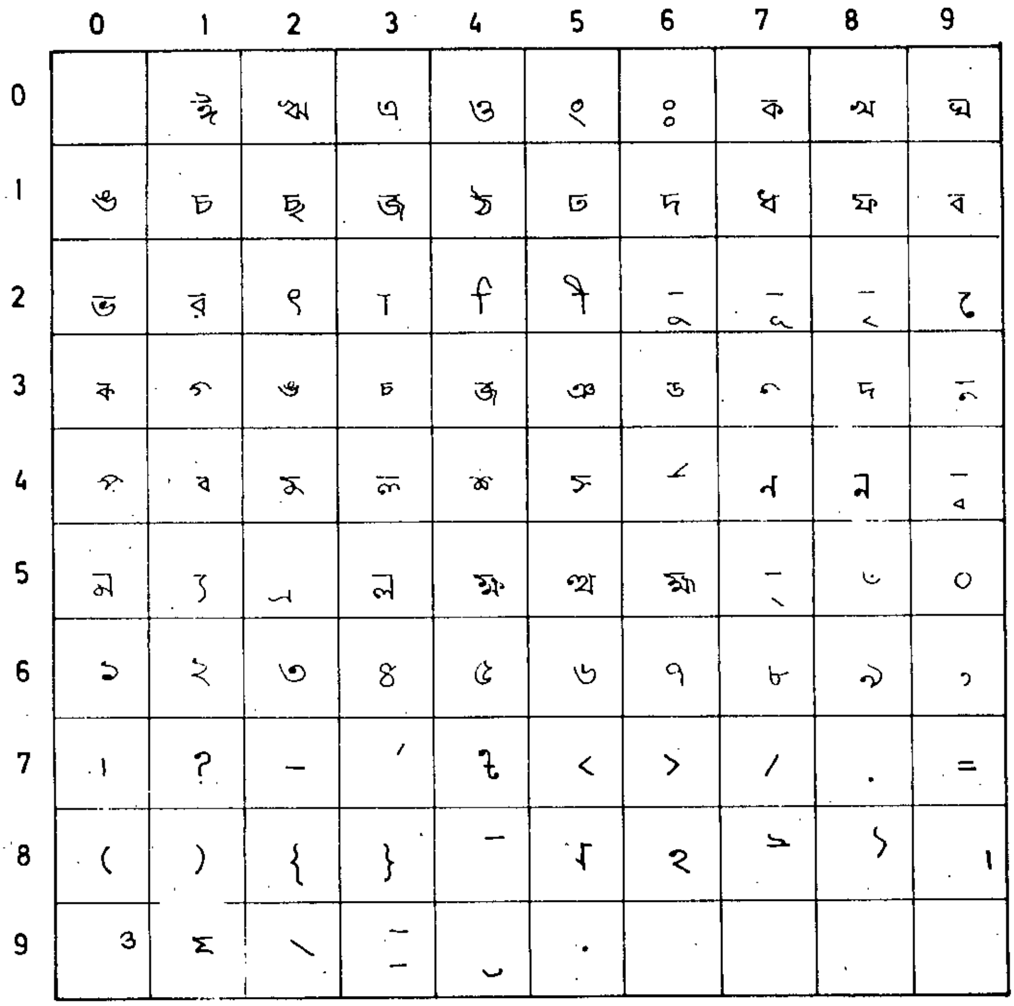
Solid font printers are widely used for bulk and letter quality printing. Line printers are used for bulk printing and daisy wheel printers are used for letter quality printing. But the 131 Bengali graphic symbol set is inconvenient to accommodate in standard daisy wheel and line printers which typically use 96 character set. Among these 96 character set, a character is dedicated for space and on closer observation, a 95 Bengali impression symbol set (BIS) can be selected by which Bengali graphic symbols can be generated if superposition is allowed. By selecting a 95 impression symbol set, existing 96 character line printers and daisy wheel printers can be modified for printing Bengali text. Selection of Bengali impression symbols (BIS) and formation of Bengali graphic symbols by them are discussed in the following article.

4.5 SELECTION OF BENGALI IMPRESSION SYMBOLS AND FORMATION OF GRAPHIC SYMBOLS IN SOLID FONT PRINTERS

On careful and closer observation on the 131 Bengali graphic symbol set, a 95 Bengali impression symbol set (BIS) has been selected which can adequetely represent 127 Bengali graphic symbols, except # , & , @ and _ , among the 131 Bengali graphic symbols. This 95 impression symbol set in given in Table 4.1.

Selection of the impression symbols has been made on the assumption that the Bengali graphic symbols are to be generated by superposition or by concatenation or by combination of superposition and concatenation of more than one impression symbols. The Bengali graphic symbols have been categorized into four groups in accordance with the number of impression symbol required for its formation, e.g., one action, where only one impression symbol will form one graphic symbol, two action where two impression symbols will form one graphic symbol either by superposition or by concatenation of the symbols, etc. Mechanism of generation of the Bengali graphic symbols by the Bengali impression symbols is shown in Table 4.2 along with the action technique needed. Among the 131 graphic symbols, # &, @ and _ can not be generated by the selected impression symbols, because the number of the impression symbols needed for other lexical and most frequent symbols can not be reduced to less than 95 in any way. The symbol র is taken as one action symbol, though it can be generated by superposing.on the ব , because the frequency of occurrence of র is extreemly high, its position is third from the most frequent graphic symbol. For accomodating the required impression symbols in a 95 symbol set, shape of some specially shaped small sized symbols of byanjana varnas and that

TABLE 4.1 BENGALI IMPRESSION SYMBOL SET (BIS)



	0	1	2	3	4	5	6	7	8	9
0		ঈ	ঋ	এ	ও	ং	ঃ	ক	ঞ	ঘ
1	ঙ	চ	ছ	জ	ঠ	ত	দ	ধ	ফ	ব
2	ভ	র	৭	া	ি	ী	ু	ূ	ৃ	ে
3	ক	ন	ঙ	চ	জ	ঞ	উ	⌒	দ	ন
4	ল	ব	স	ল	ঙ্ক	স	∠	ন	ন	ব
5	ম	ʒ	‿	ল	ঙ্ক	ত্থ	ক্ষ	↘	৺	○
6	১	২	৩	৪	৫	৬	৭	৮	৯	)
7	।	?	–	′	ৎ	<	>	/	.	=
8	(	)	{	}	¯	্	২	≻	)	।
9	3	Σ	\	=	◡	·				

(37) = ⌒ (88) =) etc.

Table 4.2: Schedule of Bengali Graphic Symbols (BGS) and Mechanism of Generation by Bengali Impression Symbols (BIS)

BGS Seq. No.	Mechanism of Generation by BIS*	Action Technique needed				BGS Seq. No.	Mechanism of Generation by BIS*	Action Technique needed			
		1	2	3	4			1	2	3	4
(01)	(62)/(84)/(85)			x		(22)	(03)/(90)		x		
(02)	(86)/(87)		x			(23)	(15)/(87)		x		
(03)	(01)	x				(24)	(14)	x			
(04)	(65)/(87)		x			(25)	(65)/(84)		x		
(05)	(65)/(87)/(94)			x		(26)	(15)	x			
(06)	(02)	x				(27)	(37)/(89)		x		
(07)	(03)	x				(28)	(62)/(84)		x		
(08)	(03)/(88)		x			(29)	(86)/(89)		x		
(09)	(04)	x				(30)	(16)	x			
(10)	(04)/(88)		x			(31)	(17)	x			
(11)	(05)	x				(32)	(39)/(89)		x		
(12)	(06)	x				(33)	(40)/(89)		x		
(13)	(07)	x				(34)	(18)	x			
(14)	(08)	x				(35)	(19)	x			
(15)	(31)/(89)		x			(36)	(20)	x			
(16)	(09)	x				(37)	(42)/(89)		x		
(17)	(10)	x				(38)	(91)/(89)		x		
(18)	(11)	x				(39)	(21)	x			
(19)	(12)	x				(40)	(43)/(89)		x		
(20)	(13)	x				(41)	(44)/(89)		x		
(21)	(19)/(85)		x			(42)	(91)/(92)/(89)			x	

Table 4.2 contd.

BGS Seq. No.	Mechanism of Generation by BIS*	Action Technique needed			
		1	2	3	4
(43)	(45)/(89)		x		
(44)	(86)/(84)		x		
(45)	(91)/(78)/(89)			x	
(46)	(65)/(84)/(78)			x	
(47)	(15)/(78)		x		
(48)	(22)	x			
(49)	(23)	x			
(50)	(24)	x			
(51)	(25)	x			
(52)	(26)	x			
(53)	(27)	x			
(54)	(28)	x			
(55)	(29)	x			
(56)	(29)/(87)		x		
(57)	(23)/(87)		x		
(58)	(30)	x			
(59)	(31)	x			
(60)	(32)	x			
(61)	(33)	x			
(62)	(34)	x			
(63)	(35)	x			
(64)	(36)	x			
(65)	(37)	x			
(66)	(38)	x			
(67)	(39)	x			
(68)	(40)	x			
(69)	(41)	x			
(70)	(42)	x			
(71)	(43)	x			
(72)	(44)	x			
(73)	(91)/(92)		x		
(74)	(45)	x			
(75)	(46)	x			
(76)	(47)	x			
(77)	(48)	x			
(78)	(49)	x			
(79)	(50)	x			
(80)	(51)	x			
(81)	(52)	x			
(82)	(53)	x			
(83)	(54)	x			
(84)	(13)/(90)		x		
(85)	(15)/(87)/(94)			x	
(86)	(04)/(84)		x		
(87)	(55)	x			
(88)	(91)/(92)/(89)/(90)				x

Table 4.2 contd.

BGS Seq. No.	Mechanism of Generation by BIS*	Action Technique needed				BGS Seq. No.	Mechanism of Generation by BIS*	Action Techniq needed			
		1	2	3	4			1	2	3	4
(89)	(56)	x				(111)	(73)	x			
(90)	(73)	x				(112)					
(91)	(57)	x				(113)	(74)	x			
(92)	(58)	x				(114)	(70)/(72)		x		
(93)	(59)	x				(115)	(77)/(92)		x		
(94)	(60)	x				(116)	(75)	x			
(95)	(61)	x				(117)	(76)	x			
(96)	(62)	x				(118)	(77)	x			
(97)	(63)	x				(119)	(95)/(77)/(78)			x	
(98)	(64)	x				(120)	(70)/(72)/(77)/(92)				x
(99)	(65)	x				(121)	(78)	x			
(100)	(66)	x				(122)	(79)	x			
(101)	(67)	x				(123)					
(102)	(68)	x				(124)					
(103)	(69)	x				(125)	(80)	x			
(104)	(69)/(95)		x			(126)	(81)	x			
(105)	(78)/(95)		x			(127)	(82)	x			
(106)	(70)	x				(128)	(83)	x			
(107)	(71)	x				(129)	(70)/(93)		x		
(108)	(70)/(78)		x			(130)	(93)/89)		x		
(109)	(72)	x				(131)					
(110)	(73) + (73)		x								

* Note : 1. (   ) means impression symbol of Table 4.1 corresponding to number enclosed.

2. / means superposition.

3. + means concatenation

4. An empty entry indicates that character can not be generated by BIS.

of the urdhacomma of Table 3.4 have to be changed, e.g.,  is to be generated as  , etc.

In the line printer, the superposition of symbols can be effected by Multipass, i.e., printing the same line with several passes. In the daisy wheel printers, the same can be effected by stop carriage where the print head is not allowed to move after printing a symbol so that the next symbol can be superposed. Line printers have an inherent drawback that all characters occupy equal horizontal space. Over and above this, there is a mandatory space between any two successive character positions. A line printer obviously can not satisfy the variable width requirement of the Bengali graphic symbols and its output would therefore be of poor typographical quality. Daisy wheel printers allow null and fine increment of carriage movement and variable width requirement can be satisfied in this device.

4.6 STATISTICAL CONSIDERATION IN SELECTION OF BENGALI IMPRESSION SYMBOLS

To reduce complexity and loss of printing speed, one should attempt to minimise the number of passes required, i.e., the number of cases where superposition will have to be effected. As the superposition cannot be eliminated, it has been attempted that the Graphic symbols which occur most frequently should have a minimum of superposition. In order to achieve this, the frequency of occurrence of the Bengali graphic symbols of Table 3.7 has been used. The frequency of occurrence of the Bengali impression symbols have been computed from Table 3.7 on the basis of 186,037 no. of occurrence and given in Table 4.3. Frequency distribution of occurrence of these impression symbols is given in Fig. 4.1.

Statistics of Bengali Graphic Symbol (BGS) generation by the Bengali Impression Symbols(BIS) is given in Table 4.4. From Group I consisting of 32 graphic symbols with individual frequency of 0.6% and above having cumulative frequency of 72.3727%, 17 graphic symbols with cumulative frequency of 46.7573% have been selected as one action symbols, 13 graphic symbols with cumulative frequency of 22.2847% have been selected as two action symbols and remaining 2 graphic symbols with cumulative frequency of 3.3307% have been selected as three action symbols. From Group II consisting of 32 graphic symbols with individual frequency of 0.1% and above but below 0.6% having cumulative frequency of 9.3726%, 25 graphic symbols with cumulative frequency of 7.4272% have been selected as one action symbols, 5 graphic symbols with cumulative frequency of 1.2538% have been selected as two action symbols and remaining 2 graphic symbols with cumulative frequency of 0.6916% have been selected as three action symbols. From Group III consisting of 67 graphic symbols with individual frequency of below 0.1% having cumulative frequency of 1.7030%, 43 graphic symbols with cumulative frequency of 1.2588% have been selected as one action symbol, 15 graphic symbols with cumulative frequency of 0.4236% have been selected as two action symbols, 3 graphic symbols with cumulative frequency of

Table 4.3: Frequency of Occurrence (on the basis of 186,037 no. of occurrence)of Bengali Impression Symbols (BIS)

BIS Seq. No.	No. of Occurrence*	% of Occurrence	BIS Seq. No.	No. of Occurrence*	% of Occurrence
(01)	741	0.3983	(23)	15,133	8.1344
(02)	5	0.0026	(24)	6,650	3.5745
(03)	1,425	0.7659	(25)	1,297	0.6971
(04)	999	0.5369	(26)	2,033	1.0927
(05)	315	0.1693	(27)	372	0.1999
(06)	86	0.0462	(28)	340	0.1827
(07)	5,328	2.8639	(29)	11,792	6.3385
(08)	806	0.4332	(30)	239	0.1284
(09)	220	0.1182	(31)	1,341	0.7208
(10)	48	0.0258	(32)	167	0.0897
(11)	1,004	0.5396	(33)	106	0.0569
(12)	1,103	0.5928	(34)	14	0.0075
(13)	1,396	0.7503	(35)	112	0.0602
(14)	304	0.1634	(36)	5	0.0026
(15)	1,379	0.7412	(37)	682	0.3665
(16)	2,404	1.2922	(38)	133	0.0714
(17)	890	0.4783	(39)	5,977	3.2128
(18)	228	0.1225	(40)	2,784	1.4964
(19)	4,503	2.4204	(41)	25	0.0134
(20)	800	0.4300	(42)	3,184	1.7114
(21)	7,575	4.0717	(43)	3,158	1.6975
(22)	224	0.1204	(44)	1,493	0.8025

0

Table 4.3 Contd.

BIS Seq. No.	No. of Occurrence*	% of Occurrence	BIS Seq. No.	No. of Occrrence*	% of Occurrence
(45)	3,351	1.8012	(72)	981	0.5273
(46)	718	0.3859	(73)	344	0.1849
(47)	2	0.0010	(74)	0	0.0000
(48)	41	0.0220	(75)	0	0.0000
(49)	295	0.1585	(76)	0	0.0000
(50)	117	0.0628	(77)	1	0.0005
(51)	1,304	0.7009	(78)	3,121	1.6776
(52)	1,388	0.7460	(79)	170	0.0913
(53)	47	0.0252	(80)	6	0.0032
(54)	309	0.1660	(81)	6	0.0032
(55)	0	0.0000	(82)	0	0.0000
(56)	6	0.0032	(83)	0	0.0000
(57)	138	0.0741	(84)	10,216	5.4913
(58)	294	0.1580	(85)	2,386	1.2825
(59)	45	0.0241	(86)	4,338	2.3317
(60)	83	0.0446	(87)	3,706	1.9920
(61)	48	0.0258	(88)	46	0.0247
(62)	7,474	4.0174	(89)	24,876	13.3715
(63)	36	0.0193	(90)	75	0.0403
(64)	49	0.0263	(91)	3,983	2.1409
(65)	1,389	0.7466	(92)	665	0.3574
(66)	29	0.0155	(93)	0	0.0000
(67)	37	0.0198	(94)	17	0.0091
(68)	38	0.0204	(95)	296	0.1591
(69)	607	0.3262	SPACE	23,286	12.5168
(70)	812	0.4364			
(71)	41	0.0220			

* Note: Computed from Table 3.7.

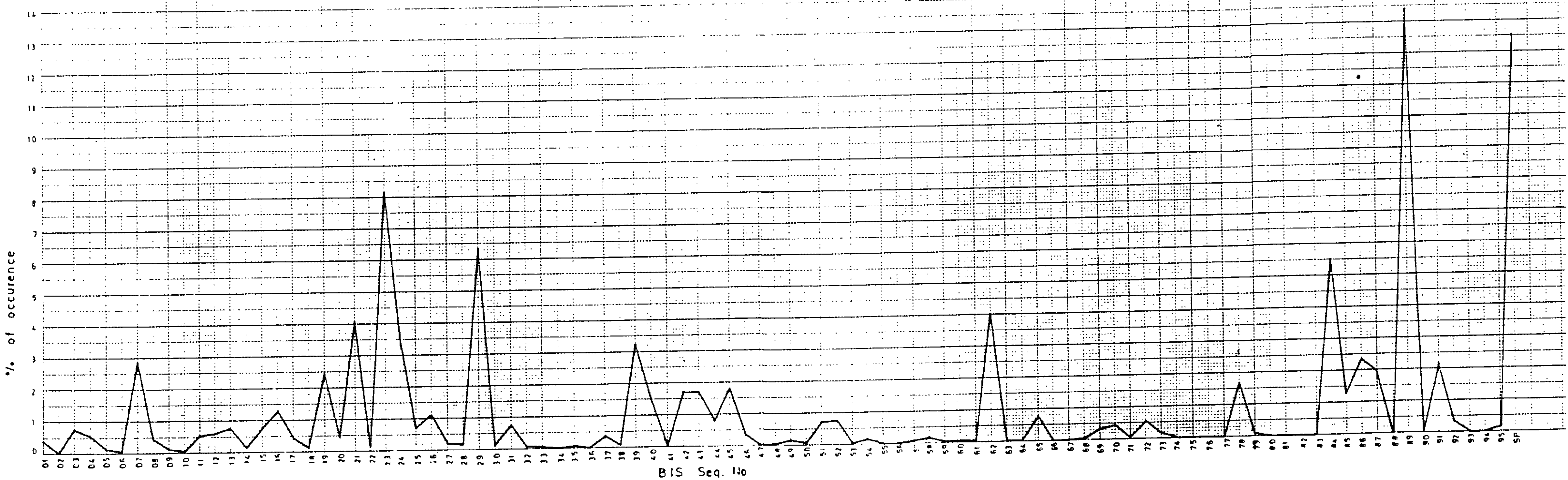


FIG. 4.1 FREQUENCY DISTRIBUTION OF OCCURENCE OF THE BENGALI IMPRESSION SYMBOL SET

Table 4.4: Statistics of Bengali Graphic Symbol (BGS) Generation By Bengali Impression Symbols (BIS)

Group (on the basis of % of occurrence)	Total No.of BGS	Cumulative freqs. in %	1 action Technique		2 action Technique		3 action Technique		4 action Technique		BGS not be Generated	
			Total No.of BGS	Cumul. freqs. in %	Total No.of BGS	Cumul. freqs. in %	Total No.of BGS	Cumul. freqs. in %	Total No.of BGS	Cumul. freqs. in %	Total No.of BGS	Cumul. freqs. in %
I (0.6% & above)	32	72.3727	17	46.7573	13	22.2847	2	3.3307	0	0.0000	0	0.0000
II (0.1% & above but below 0.6%)	32	9.3726	25	7.4272	5	1.2538	2	0.6916	0	0.0000	0	0.0000
III (Below 0.1%)	67	1.7030	43	1.2588	15	0.4236	3	0.0128	2	0.0078	4	0.0000
Total	131	83.4483	85	55.4433	33	23.9621	7	4.0351	2	0.0078	4	0.0000
SPACE		16.5515										

0.0128% have been selected as three action symbols, 2 graphic symbols with cumulative frequency of 0.0078% have been selected as four action symbols and remaining 4 graphic symbols with cumulative frequency of 0.0000% are not to be generated by impression symbols. This results to selection of 85 graphic symbols with cumulative frequency of 55.4433% as one action symbols, 33 graphic symbols with cumulative frequency of 23.9621% as two action symbols, 7 graphic symbols with cumulative frequency of 4.035% as three action symbols, 2 graphic symbols with cumulative frequency of 0.0078% as four action symbols and remaining 4 graphic symbols with cumulative frequency of 0.000% are not to be generated by impression symbols.

This selection of impression symbols will result to

Average No. of Superposition/Graphic Symbol,

$$S_{av} = \frac{\sum_{i=1}^{4} i,c_i}{\sum_{i=1}^{4} c_i}$$

$$= \frac{1 \times 71.9948 + 2 \times 23.9621 + 3 \times 4.0351 + 4 \times 0.0078}{71.9948 + 23.9621 + 4.0351 + 0.0078}$$

$$= 1.3205$$

where, S_{av} = average no. of superposition/ graphic symbol

C_i = % cumulative frequency of ith action symbols

i = 1 for 1 action symbols & space

= 2,3,4 for 2 action, 3 action, 4 action symbols respectivel.

Therefore, the speed penalty in case of daisy wheel printer will not be much pessimistic.

Computation of speed penalty for line printer is more difficult. If a line printer is assumed with 132 graphic symbols line, as the cumulative frequency of 2 action symbols is 23.9621%, 31.6299 symbols out of 132 will need a second pass, i.e., a second pass is mandatory for all lines. As the cumulative frequency of 3 action symbols is 4.0351%, 5.3263 symbols out of 132 will need a third pass, i.e., a third pass is, also, mandatory for all lines. As the cumulative frequency of 4 action symbols is 0.0078%, 0.0103 symbols out of 132 will need a fourth pass, i.e., 103 lines out of 10,000 lines would need a fourth pass. Thus average number of pass would be

$$\frac{3 \times 10{,}000 + 103}{10{,}000} = 3.0103.$$

CHAPTER - 5

ENCODING BENGALI INFORMATION PRIMITIVES

5.1 INTRODUCTION

The 131 Bengali Graphic Symbol Set (BGS) of Table 3.4 has been selected for entering text into computer through key board and the same will be used for Information Interchange, i.e., the same symbols are selected as Bengali Information Primitives. The Information primiset should also contain appropriate control code for computer usage. Numeric codes for these information primitives are needed for machine representation and data communication over remote places.

5.2 CODING SCHEME

Along with the 131 Bengali Graphic Symbols, two more characters are needed as information primitives, they are Space and Delete characters. Space is needed for inserting space between two consecutive words and Delete character is needed for indicating the character which is deleted from the record.

The 32 industry standard ASCII control codes (appendix D) have been taken as control codes for the present coding scheme. With 32 control codes, Space and Delete character, the total nos. of Bengali Information primitives become 165. These 165 Information primitives will need ld 165=7.3663, i.e., 8 bits for encoding.

If a parity bit is used for error checking, the number of bits required will be 9. The Inlet 8251A Programmable Communication Interface[12] can handle 5-8 bit characters by inserting an additional parity bit for serial data communication. This 8-bit coding scheme for 165 Bengali Information Primitives can be handled by the Intel 8251A for serial data communication. But most of the serial data communication systems handle a 7-bit character with an additional parity bit resulting the total bits required is 8. On the other hand, for parallel data communication and machine representation, this 8-bit coding scheme with an additional parity bit can not be used, because no fractional byte can be represented in a typical computer system. In this case 2 bytes, i.e., 16 bits are to be used for representing 165 Bengali Information Primitives with a parity bit, which will obviously be inefficient from the view point of computer resource requirement, i.e., more internal memory will be required for machine representation.

To make the coding scheme efficient, it has been dicided that an 8-bit coding scheme will be used for machine representation. For error checking in serial and parallel data communication, a special code mapping is to be used for inserting parity bit as discussed in the next article. For these special code mapping scheme, another information primitive, i.e., CXT (Code eX Tender) is required.

All these 166 information primitives are encoded in an 8-bit coding scheme and named the BCII code, i.e., Bengali Code for Information Interchange. These codes are given in Table 5.1. Control codes, Space and numerals are kept ASCII compatible. The lexical symbols are encoded in their lexical order of Table 3.4 such that their lexical ordering can be analyzed by numeric analysis. Code Space 80 Hex to 9F Hex is kept unused to provide future provision of extending control codes if needed. DELete character is encoded as FF Hex and Code eXTender (CXT) is encoded as 7F Hex

5.3 ERROR CHECKING IN DATA COMMUNICATION

The normal practice of error checking in either parallel or serial data communication is to use a parity bit in addition with the character bits. Normally the additional parity bit is placed at the most significant position of the code, i.e., with a 7-bit coding scheme, the additional parity bit is placed at the 7th bit and the 7-bit character is placed from the 6th to 0th bits of an 8-bit code. As the BCII coding scheme employs 8-bits for encoding the information primitives, an additional parity bit can be inserted with the 8-bit character which can easily be handled by the Intel 8251A Programmable Communication Interface.

Table 5.1: Bengali Code For Information Interchange (BCII)

Code (Hex)	Char	Code (Hex)	Char	Code (Hex)	Char	Code (Hex)	Char
00	NUL	16	SYN	2C	ঃ	42	খ
01	SOH	17	ETB	2D	,	43	গ
02	STX	18	CAN	2E	্	44	ঘ
03	ETX	19	EM	2F	ঁ	45	ঙ
04	EOT	1A	SUB	30	০	46	চ
05	ENQ	1B	ESC	31	১	47	ছ
06	ACK	1C	FS	32	২	48	জ
07	BEL	1D	GS	33	৩	49	ঝ
08	BS	1E	RS	34	৪	4A	ঞ
09	HT	1F	US	35	৫	4B	ট
0A	LF	20	SP	36	৬	4C	ঠ
0B	VT	21	অ	37	৭	4D	ড
0C	FF	22	ই	38	৮	4E	ঢ
0D	CR	23	ঈ	39	৯	4F	ণ
0E	SO	24	উ	3A	’	50	ত
0F	SI	25	ঊ	3B	;	51	থ
10	DLE	26	ঋ	3C	:	52	দ
11	DC1	27	এ	3D	।	53	ধ
12	DC2	28	ঐ	3E	?	54	ন
13	DC3	29	ও	3F	!	55	প
14	DC4	2A	ঔ	40	–	56	ফ
15	NAK	2B	ং	41	ক	57	ব

Table 5.1 Contd.

Code (Hex)	Char	Code (Hex)	Char	Code (Hex)	Char	Code (Hex)	Char
58	ভ	6F	্ন	86		9D	
59	স	70	্ত	87		9E	
5A	য	71	চ	88		9F	
5B	র	72	ঞ	89		A0	∠
5C	ল	73	ঞ্চ	8A		A1	ⅆ
5D	ক্ষা	74	উ	8B		A2	ন
5E	ষ	75	ঁ	8C		A3	ৰ
5F	স	76	ন	8D		A4	ন
60	হ	77	ন	8E		A5	ƺ
61	য়	78	ক	8F		A6	৴
62	ড়	79	ব	90		A7	ল
63	ঢ়	7A	ম	91		A8	স্ফ
64	ৎ	7B	ল	92		A9	ঙ্গ
65	T	7C	[illegible]	93		AA	ট্র
66	ি	7D	ত	94		AB	ভ
67	ী	7E	স	95		AC	ল্ল
68	ু	7F	CXT	96		AD	ঞ্জ
69	ূ	80		97		AE	ঙ্ক
6A	ৃ	81		98		AF	//
6B	ে	82		99		B0	/
6C	ৈ	83		9A		B1	#
6D	ৗ	84		9B		B2	ʅ
6E	ক	85		9C		B3	+

Table 5.1 Contd.

Code (Hex)	Char	Code (Hex)	Char	Code (Hex)	Char	Code (Hex)	Char
B4	×	C7		DA		ED	
B5	<	C8		DB		EE	
B6	>	C9		DC		EF	
B7	/	CA		DD		F0	
B8	%	CB		DE		F1	
B9	*	CC		DF		F2	
BA	.	CD		E0		F3	
BB	=	CE		E1		F4	
BC	&	CF		E2		F5	
BD	@	D0		E3		F6	
BE	(	D1		E4		F7	
BF	)	D2		E5		F8	
C0	{	D3		E6		F9	
C1	}	D4		E7		FA	
C2	[	D5		E8		FB	
C3	]	D6		E9		FC	
C4	−	D7		EA		FD	
C5		D8		EB		FE	
C6		D9		EC		FF	DEL

As most of the typical serial data communication systems handle a 7-bit character and insert an additional parity bit at the 7th bit of the 8-bit code, a code mapping scheme is proposed for these cases where the 7th bit of an 8-bit code will be used as parity bit and the CXT code extender will be used dor extending the codes.

The BCII codes are divided into two groups- the Ist group with characters having codes from OOHex to 7F Hex, i.e., with the 7th bit O(zero) and the 2nd group with characters having codes from 80 Hex to FF Hex, i.e., with the 7th bit 1. For both the groups, in this code mapping scheme, parity bit is to be set at the 7th bit and the 6th to Oth bits are to be transmitted as contained in the actual BCII code. For differentiating the code group, a prefix CXT code byte is to be transmitted with appropriate parity set at the 7th bit with the codes of the 2nd group for indicating the code extention. At the receiving end, the 7th bit of the code following a CXT code is to be set to 1 and other bits, i.e., the 6th to Oth bits are to be restored as transmitted. In other cases, where the code is not preceded by a CXT code, the 7th bit is to be set to O (zero) and the 6th to Oth bits are to be restored as transmitted. The algorithms of code mapping at both the transmitting and receiving end of a serial data communication system are given in Fig. 5.1 and 5.2 respectively.

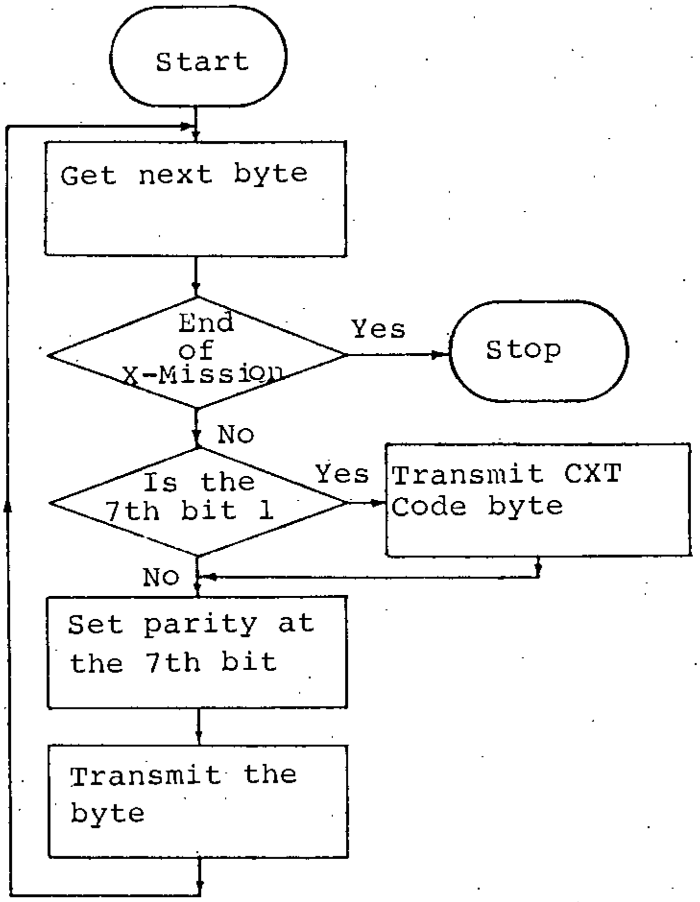


Fig. 5.1: Algorithm of Code Mapping At the Transmitting End.

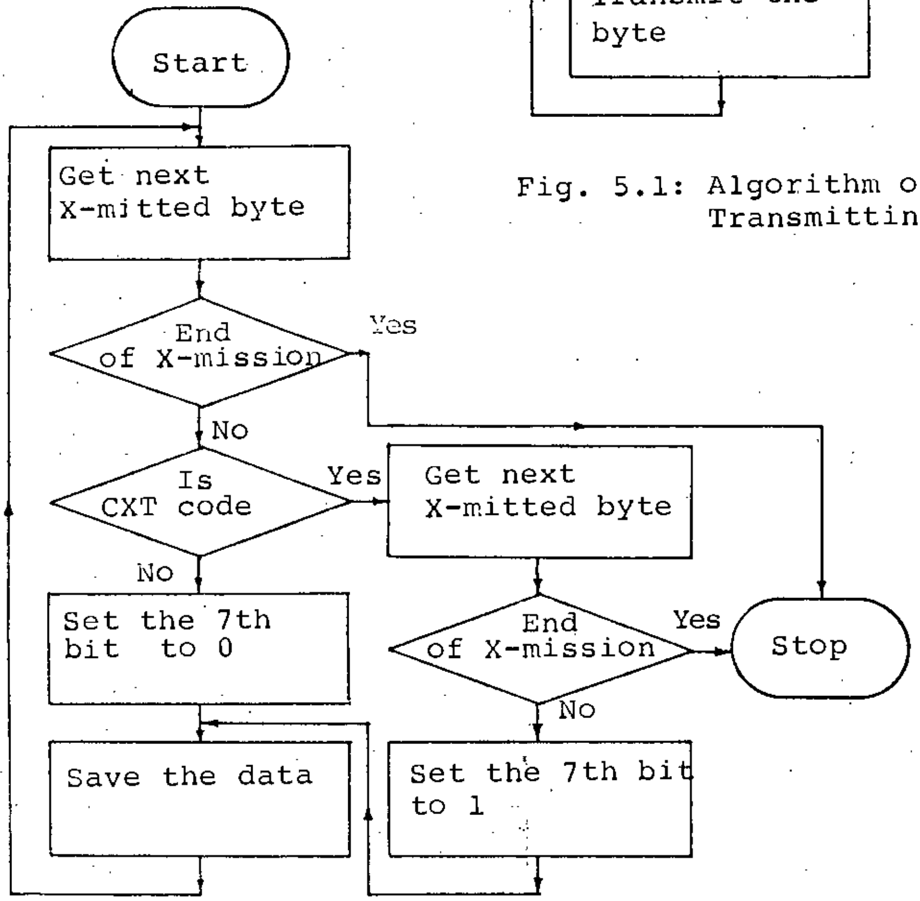


Fig. 5.2: Algorithm of Code Mapping at the Receiving End.

For parallel data communication by 8 parallel lines, the same code mapping scheme is to be used as in serial data communication.

5.4 STATISTICAL CONSIDERATION IN CODING SCHEME

To reduce the average bit required per information primitive, i.e., to reduce the cases where the CXT code is required to be transmitted, 38 least frequent information primitives are encoded in the 2nd group of codes, i.e., in the code space 80 Hex to FF Hex and the other 128 most frequent and ASCII compatible information primitives are encoded in the Ist group of codes, i.e., in the code space 00 Hex to 7F Hex. The cumulative frequency of the Ist group codes is 96.4800% and that of the 2nd group codes is 3.5193%. This leads to

Average Bits/Information Primitive,

$$B_{av} = \frac{\sum_{i=1}^{2} b_i . c_i}{\sum_{i=1}^{2} c_i} = \frac{8 \times 96.4800 + 16 \times 3.5193}{96.4800 + 3.5193}$$

$$= 8.2815$$

where, B_{av} = average bits/information primitive

b_i = bits/information primitive for group i

c_i = % cumulative frequency of information primitives of group i

i = 1 for the 1st group of codes

= 2 for the 2nd group of codes.

Therefore, in this code mapping scheme, average bit required per information primitive is less than 9, which would be required when the 8-bit BCII codes are to be serially transmitted with an additional parity bit.

CHAPTER 6

ADAPTATION OF IBM PC KEY-BOARD AS BENGALI KEY-BOARD

6.1 INTRODUCTION

The Adapted BCII key-board lay-out of Fig. 3.6 has been deviced with the aim to change the existing microcomputer key-board to Bengali key-board for using with the same computer. For this purpose, the widely used IBM PC microcomputer has been selected, because a number of microcomputers manufactured by other manufacturers are available which are compatible with the IBM PC. The selection of IBM PC will provide the opportunity of easy implementation of the developed key-board handler routine with a number of widely used microcomputers.

Key-board handler routines for generating BCII code from the IBM PC key-board have been developed and discussed in this chapter.

6.2 BRIEF DESCRIPTION OF IBM PC KEY-BOARD

The IBM PC key-board has 83 keys and the key-board is divided into three zones according to key function-

- zone 1: Typewriter keys and Control key
- zone 2: Numeric keypad
- zone 3: Function keys.

Lay-out of the key-board is shown in Fig. 6.1.

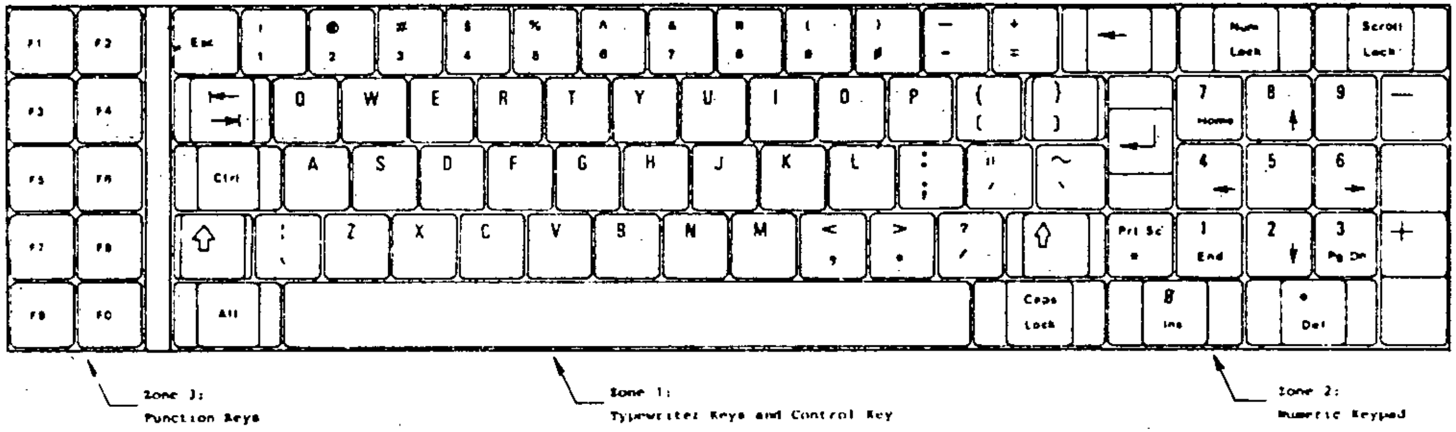


Fig. 6.1 IBM PC key-Board Lay-out

The standard typewriter keys are located in the center of the key-board, i.e., in zone 1. This zone contains 47 alphanumeric keys, 1 space bar and 10 control keys. The numeric key pad (zone 2) is on the right hand side of the key-board. The keys can be used to move the curser, or produce numbers. In the NUM LOCK "off" position, pressing the keys moves the curser according to the direction on the key, in the NUM LOCK "on" position the numeric keypad functions like a calculator keypad. This zone contains 15 keys including some special control keys. The function keys (zone 3) are located on the left-hand side of the key board. These keys can have different functions for different programs. This zone contains 10 function keys.

The key-board is an interrupt driven key-board. When a key is pressed, the key-board processor serially sends the hexa decimal scan code (location value) of the pressed key. A hardware interface circuit on the microcomputer main board receives the serial data and then converts the serial data into paralled data. When the data output register of the interface hardware is ready, it produces an interrupt request to 8259 programmable interrupt controller[12]. The 8259 then produces a type 9 interrupt request. The interrupt handler routine pointed to by the interrupt vector at location 0024H, i.e., the interrupt vector of type 9 interrupt, reads the key-board data from the data output register of the interface hardware and produces the corresponding ASCII code for final processing. When a key is released from its pressed position, the same phenomenon occures but the most significant bit of the scan code is set to 1.

If a key is kept at pressed condition for a long time, the key-board processor gives a delay and then sends the same scan code repeatedly until the key is released. When more than one keys are pressed simultaneously, the key-board processor accepts the last key pressed and sends scan code of that key ignoring all other pressed keys.

The address of the Operation Control Word 1 (OCW1) of the 8259 is 21H and that of the Operation Control Word 2 (OCW2) and Operation Control Word 3 (OCW3) is 20H. The address of the data output register of the interface hardware is 60H.

6.3 CONVERSION OF IBM PC KEY-BOARD TO ADAPTED BCII KEY-BOARD

The key-board lay-out of ABCII key-board has been deviced with 47 main keys which correspond to the 47 alphanumeric keys of zone 1 of IBM PC key-board. 131 Bengali Graphic Symbols have been placed on these 47 keys according to the ABCII key-board lay-out. Space bar is kept unchanced. Two shift keys of this zone have been used as shift 1 keys of ABCII key-board. 'Alt' and "Caps Lock" keys have been used as shift 2 keys. Other 6 control keys of this zone have been kept unchanged. 15 keys of zone 2 and 10 keys of zone 3 have been kept unchanged. Lay-out of this Adapted BCII key-board is shown in Fig. 6.2.

Each key of the Adapted BCII key-board produces 5 types of code depending upon the control key pressed along with that

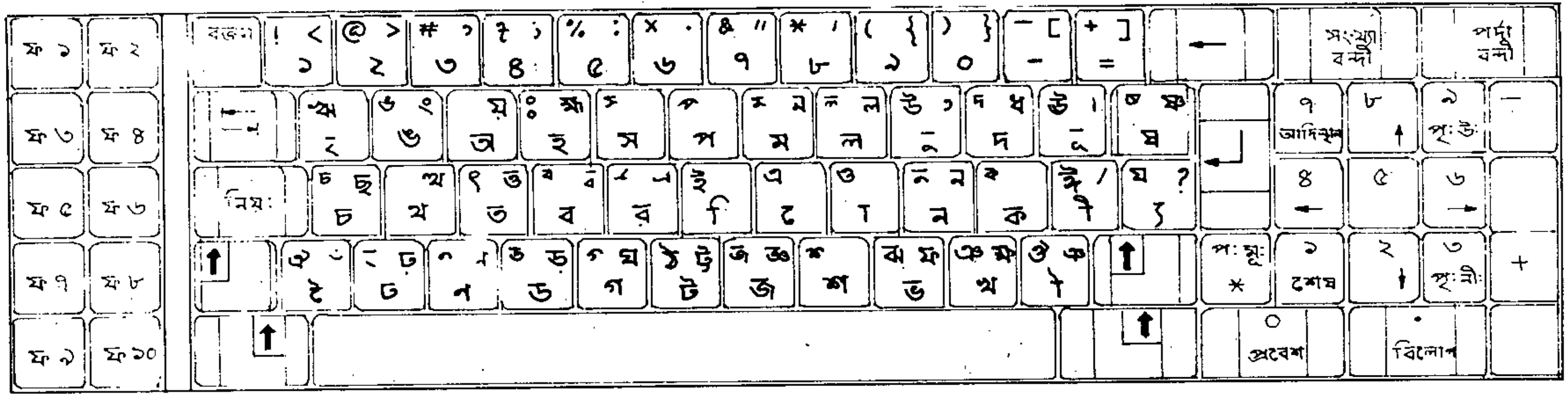


FIG. 6.2 ADAPTED BCII KEY-BOARD LAY-OUT

key. If a single key is pressed, it will produce BCII code corresponding to the symbol assigned to the normal position of that key. If either of shift 1 keys is pressed with any key, the key will produce BCII code corresponding to the symbol assigned to the shift 1 position of that key. If either of shift 2 keys is pressed with any key, the key will produce BCII code corresponding to the symbol assigned to the shift 2 position of that key. If নিয়ঃ (ctrl) key is pressed with another key, the key will produce control code assigned to that key. If সংখ্যাবন্দী(Num Lock) key is pressed with any key of the numeric key pad, the key will produce BCII code corresponding to the numeral assigned to that key. If no symbol is assigned to either of the shift positions or if any combination of either নিয়ঃ or সংখ্যাবন্দী keys with any other key is undefined, that combination of control key and other key will produce NULL (OOH) code.

The generated BCII code, for all 5 types of combination, corresponding to each key of the ABCII key-board along with the corresponding key-board scan code is shown in Table 6.1. The don't care (xx) value of the table indicates that, for the combination of that control key with the corresponding key, space has been kept in the developed routine such that user program dependable code can be generated from that key combination after inserting the desired code at that place of the code table of the developed routine.

Table 6.1: Generated BCII Code From ABCII Key-Board

Key	Key-board Scan code (Hex) Press	Release	Geneated BCII Code (Hex) Normal	Shift 1	Shift 2	Ctrl	Num Lock
বর্জন	01	81	1B	00	00	00	00
! ১ <	02	82	31	3F	B5	xx	00
@ ২ >	03	83	32	BD	B6	xx	00
# ৩ ,	04	84	33	B1	3A	xx	00
ঽ ৪ ;	05	85	34	B2	3B	xx	00
% ৫ :	06	86	35	B8	3C	xx	00
× ৬ .	07	87	36	B4	BA	xx	00
& ৭ "	08	88	37	BC	AF	xx	00
* ৮ /	09	89	38	B9	B0	xx	00
(৯ {	0A	8A	39	BE	C0	xx	00
) ০ }	0B	8B	30	BF	C1	xx	00
− _ [	0C	8C	40	C4	C2	xx	00
+ =]	0D	8D	BB	B3	C3	xx	00
←	0E	8E	08	08	08	xx	00
⇤ ⇥	0F	8F	09	00	00	00	00
ঋ ৎ	10	90	6A	26	00	xx	00
ঔ ও ং	11	91	45	70	2B	xx	00
অ য়	12	92	21	00	61	xx	00
ঃ হ ঈ	13	93	60	2C	AE	xx	00
ফ ন	14	94	5F	7E	00	xx	00
ক প	15	95	55	78	00	xx	00
ক ম য	16	96	59	7A	A4	xx	00

Table 6.1 Contd.

Key	Key-board scan code (Hex)		Generated BCII Code (Hex)				
	Press	Release	Normal	Shift 1	Shift 2	Ctrl	Num Lock
ল	17	97	5C	7B	A7	xx	00
উ ু	18	98	68	24	2D	xx	00
দ ধ দ	19	99	52	76	53	xx	00
ঊ ূ ।	1A	9A	69	25	3D	xx	00
ড ক্ষ য	1B	9B	5E	7D	AD	xx	00
↵	1C	9C	OD	00	00	00	00
নিয়:	1D	9D	00	00	00	00	00
চ ছ	1E	9E	46	71	47	xx	00
ঝ ঋ	1F	9F	51	00	AC	xx	00
ৎ ত ও	20	A0	50	64	AB	xx	00
ব ব ব	21	A1	57	79	A3	xx	00
র	22	A2	5B	A0	A6	xx	00
ই ি	23	A3	66	22	00	xx	00
এ ে	24	A4	6B	27	00	xx	00
ও া	25	A5	65	29	00	xx	00
ন ন ন	26	A6	54	77	A2	xx	00
ক ক	27	A7	41	6E	00	xx	00
ঈ ী /	28	A8	67	23	B7	xx	00
ম ১ ?	29	A9	A5	5A	3E	xx	00
⇧	2A	AA	00	00	00	00	00
ং ঃ ঁ	2B	AB	6C	28	2F	xx	00
ট ঢ	2C	AC	4E	2E	63	xx	00
ণ ন	2D	AD	4F	75	A1	xx	00

Table 6.1 Contd.

Key	Key-board scan code (Hex)		Generated BCII Code (Hex)				
	Press	Release	Normal	Shift 1	Shift 2	Ctrl.	Num Lock
ঢ ড ড়	2E	AE	4D	74	62	xx	00
ৃ গ ঘ	2F	AF	43	6F	44	xx	00
ঠ ট ট্ট	30	B0	4B	4C	AA	xx	00
অ জ জ্ঞ	31	B1	48	72	A9	xx	00
ক্ষ শা	32	B2	5D	7C	00	xx	00
ঝ ভ ফ	33	B3	58	49	56	xx	00
ঞ খ ক্ষ	34	B4	42	4A	A8	xx	00
ঔ ৗ ক্র	35	B5	6D	2A	73	xx	00
⇧ ⇧	36	B6	00	00	00	00	00
প: * মূ:	37	B7	B9	xx	xx	xx	00
⇧	38	B8	00	00	00	00	00
ফাকা	39	B9	20	20	20	xx	00
⇧	3A	BA	00	00	00	00	00
ফ-১	3B	BB	xx	00	00	00	00
ফ ২	3C	BC	xx	00	00	00	00
ফ ৩	3D	BD	xx	00	00	00	00
ফ ৪	3E	BE	xx	00	00	00	00
ফ ৫	3F	BF	xx	00	00	00	00
ফ ৬	40	C0	xx	00	00	00	00
ফ ৭	41	C1	xx	00	00	00	00
ফ ৮	42	C2	xx	00	00	00	00
ফ ৯	43	C3	xx	00	00	00	00

Table 6.1 Contd.

Key	Key-board scan code (Hex)		Generated BCII Code (Hex)				
	Press	Release	Normal	Shift 1	Shift 2	Ctrl.	Num Lock
ফ ১০	44	C4	xx	00	00	00	00
সংখ্যা বন্দী	45	C5	00	00	00	00	00
পর্দা বন্দী	46	C6	xx	xx	xx	xx	00
৭ আদিস্থান	47	C7	00	00	00	0B	37
৮ ↑	48	C8	1E	00	00	xx	38
৯ পৃঃ উঃ	49	C9	xx	xx	xx	xx	39
—	4A	CA	40	40	40	xx	40
৪ ←	4B	CB	1D	xx	xx	xx	38
৫	4C	CC	00	00	00	xx	35
৬ →	4D	CD	1C	xx	xx	xx	36
+	4E	CE	B3	B3	B3	xx	B3
১ শেষ	4F	CF	xx	xx	xx	xx	31
২ ↓	50	D0	1F	xx	xx	xx	32
৩ পৃঃ নীঃ	51	D1	xx	xx	xx	xx	33
০ প্রবেশ	52	D2	xx	xx	xx	xx	30
. বিলোপ	53	D3	FF	FF	FF	xx	BA

Key-board handling routines for generating BCII code from the adapted BCII key-board have been developed and discussed in the following articles.

6.4 DEVELOPMENT OF BENGALI KEY-BOARD HANDLING ROUTINES

Though the IBM PC key-board is an interrupt driven key-board, the key-board can also be handled in the non-interrupting mode by masking out the key-board interrupt by the Interrupt Mask Register (IMR) programmed through the Operation Control Word 1 (OCW1) of the 8259 PIC. The key-board data can be polled from the data output register of the interface hardware by checking the key-board interrupt request in the Interrupt Request Register(IRR) via the Operation Control Word 3(OCW3) of the 8259 PIC. This non-interrupting mode key-board handler can be used in small and dedicated applications. But for the large, generalized and flexible applications, interrupting mode key-board handler is essential.

Software routines for both the non-interrupting and interrupting mode key-board handling have been developed and discussed in the following articles.

6.5 NON-INTERRUPTING MODE KEY-BOARD HANDLING ROUTINE

The non-interrupting mode key-board handling routine has been written in BASIC[13] and compiled[14] and linked[15] to produce an executable file such that the executable file can be loaded under System[16] for execution. The routine contains four portions whose algorithm is discussed seperately in the following articles and the program listing of the routine is given in Table 6.2.

6.5.1 Initialization

The initialization is the first step of the non-interrupting mode key-board handling routine. This portion first reads code tables in the memory and then mask outs the key-board interrupt by the IMR programmed through the OCW1 of the 8259 PIC. The key-board interrupt request line is connected to the IR1 line of the Interrupt Request Register (IRR) of the 8259 PIC. An 1 at the first bit of the OCW1, when programed, mask outs the key-board interrupt. This initialization portion then sends read IRR on next $\overline{RD}$ pulse command by writting 0AH in the OCW3. After initialization, the control is transferred to the code generation portion of the key-board handling routine. Algorithm of the initialization process is given in Fig. 6.3a.

6.5.2 Key-Board Data Reading

Key-board data reading process is a subroutine called by the code generation portion of the non-interrupting mode key-board handling routine. This subroutine first toggles the clock at the 7th bit of the port 61H and then tests the presence of the key-board interrupt request in the Interrupt Request

Register (IRR) via the Operation Control Word 3 (OCW3) of the 8259 PIC. An 1 at the first bit of the OCW3 indicateds that the key-board interrupt request is present in the IRR. This presence of key-board interrupt request indicates that key-board data is ready at the data output register of the interface hardware. When the data output register gets ready, the subroutine reads the key-board data from the data output register (port 60H) and then control is transferred back to the calling routine. Algorithm of this key-board data reading subroutine is given in Fig. 6.3b.

6.5.3 Code Generation

Code generation process is the main process of the non-interrupting mode key-board handling routine. This portion first calls the key-board data reading subroutine and then generates the BCII code corresponding to the key-pressed. It first tests whether the key-board data is key-released data or not. If it is key-released data, the control is transferred to the starting of the process. If not, it tests for the shift 1, shift 2, Ctrl and Num Lock key pressed. If either of these control keys is pressed, the control is transferred to the corresponding portion of the process. If not, it generates the BCII code corresponding to the symbol assigned to the normal position of the pressed key from the normal group of code table. This process then calls the processing routine and control is transferred to the starting of the code generation process after returning from the processing subroutine.

The portion corresponding to the shift 1 key- pressed of this process first calls the key-board data reading subroutine and then tests whether the shift 1 key is released or not. If the shift 1 key is released, the control is transferred to the starting of the code generation process. If not, it tests for the shift 1 key press. If shift 1 key is pressed, the control is transferred to the starting of the shift 1 key pressed portion. If not, it tests for the key release. If the pressed key is released, the control is transferred to the starting of the shift 1 key pressed portion. If not, the process generates the BCII code corresponding to the symbol assigned to the shift 1 position of the pressed key from the shift 1 group of the code table. The process then calls the processing subroutine and control is transferred to the starting of the shift 1 key pressed portion after returning from the processing subroutine.

The portion corresponding to the shift 2 key pressed and that corresponding to the ctrl key pressed of this process do tasks similar to that of the shift 1 key pressed portion of this process.

The portion corresponding to the num lock key pressed of this process first calls the key-board data reading subroutine and then tests whether the num lock key is pressed again or not, because the num lock key is a toggle key in nature. If the num lock key is pressed again, the control is transferred to the starting of the code generation process. If not, it tests for the num lock key release. If num lock key is released, control is transferred to the starting of the num lock pressed portion. If not, it tests for the key release. If the pressed key is released, the control is transferred to the starting of the num lock key pressed portion. If not, the process generates the BCII code corresponding to the numerals assigned to the pressed keypad key from the num lock group of the code table. The process then calls the processing subroutine and control is transferred to the starting of the num lock key pressed portion after returning from the processing subroutine.

The algorithm of the code generation process is given in Fig. 6.3c.

6.5.4 Processing

The processing process is a subroutine called by the code generation process for processing the generated BCII code for whatever processing needed by the application program. For the demonstration purpose of the present work, the generated BCII code is printed in the screen. The algorithm of this demonstration processing is given in Fig. 6.3d.

6.6 INTERRUPTING MODE KEY-BOARD HANDLING ROUTINE

The interrupting mode key-board handling routine consists of a main routine or processing routine written in BASIC, two

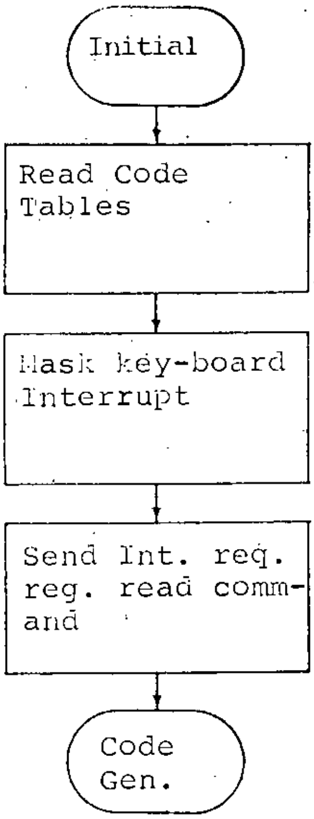


Fig. 6.3a: Algorithm of Initialization In Non-Interrupting Mode.

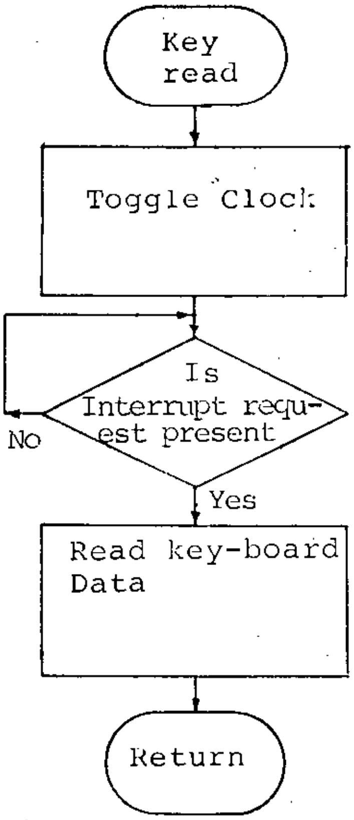


Fig. 6.3b: Algorithm of Reading Key-board Data In Non-Interrupting Mode.

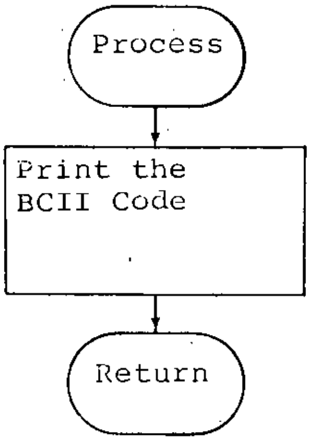


Fig. 6.3d. Algorithm of Printing BCII Code in the Screan.

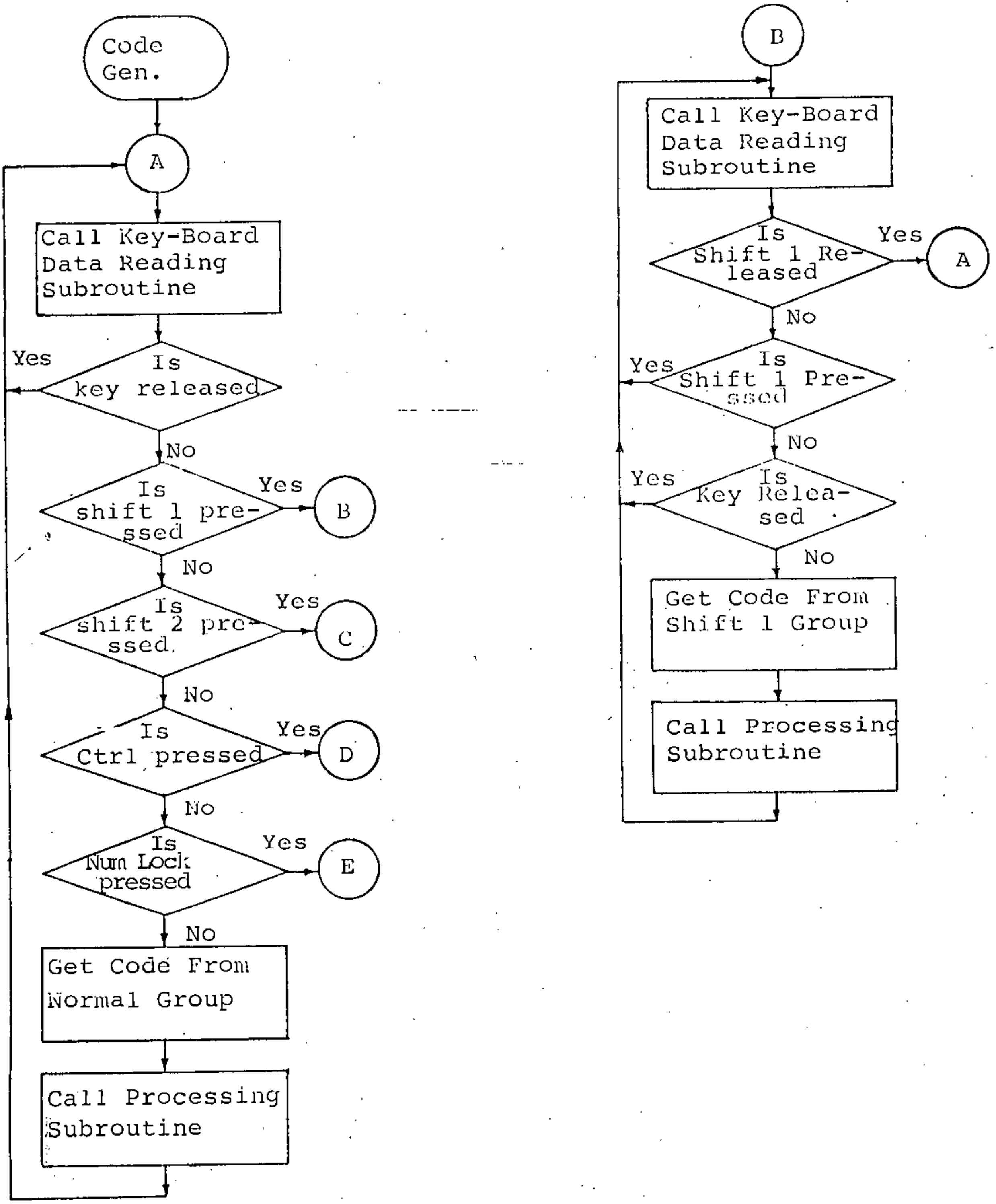


Fig. 6.3c: Algorithm of Code Generation in Non-Interrupting Mode.

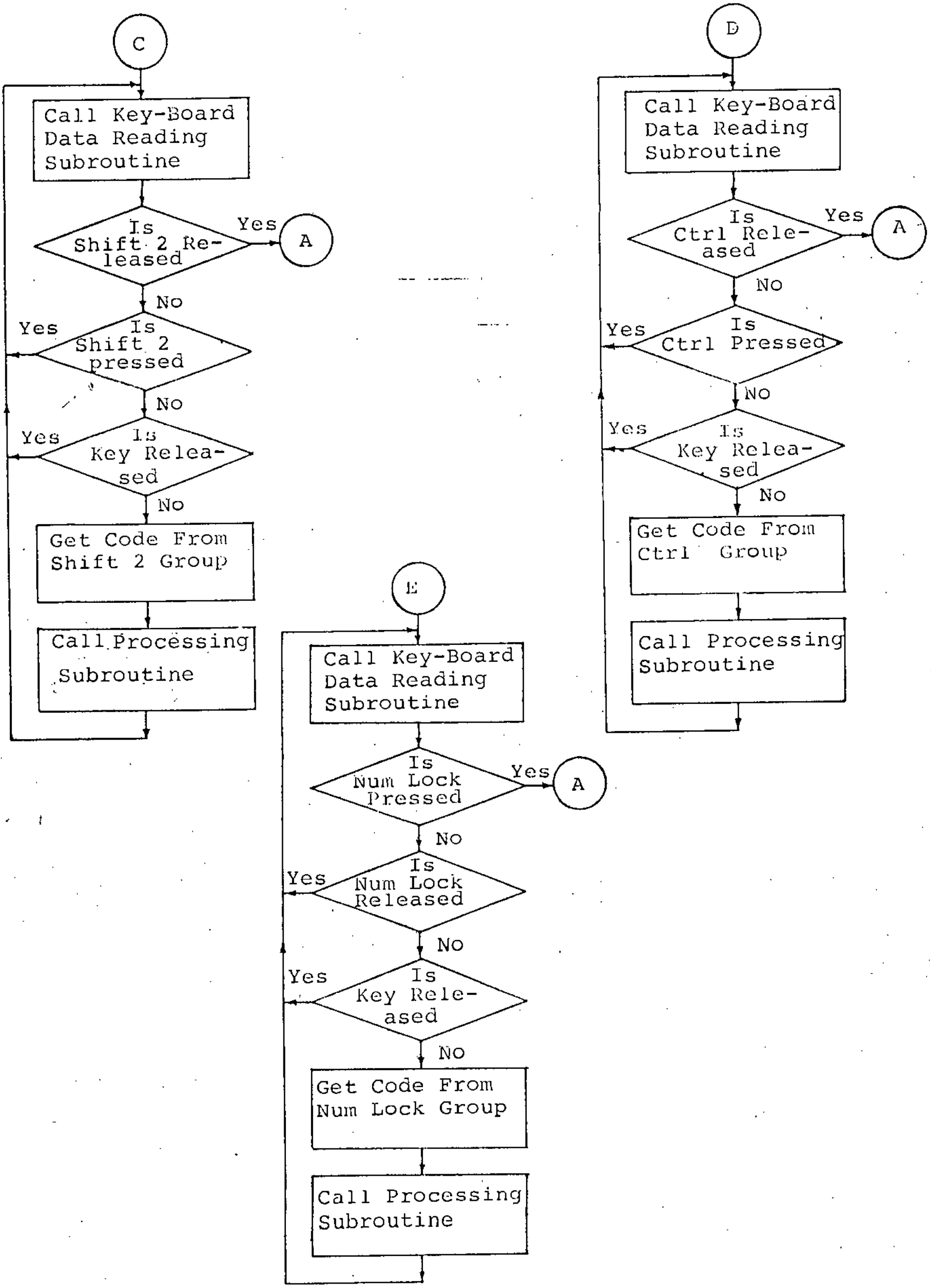


Fig. 6.3c Contd.

Table 6.2: Program Listing of Non-Interrupting Mode Key-Board Handling Routine.

```
                                                                PAGE   1
                                                                10-05-86
                                                                15:45:17
Offset  Data    Source Line         IBM Personal Computer BASIC Compiler V1.00

 001A   0002    5 REM $pagesize:54
 001A   0002    6 REM $linesize:70
 001A   0002    10 DEFINT A-Z
 001A   0002    12 REM initialization
 001A   0002    15 REM read code tables
 001A   0002    20 DIM CA(4,83)
 001A   0002    30 FOR I=0 TO 4:FOR J=1 TO 83
 0036   034A    40 READ CA(I,J)
 004A   034E    50 NEXT J,I
 0066   034E    55 REM mask keyboard interrupt
 0066   034E    60 OUT &H21,&H2
 006D   034E    65 REM send interrupt request register read command
 006D   034E    70 OUT &H20,&HA
 0074   034E    72 REM code generation
 0074   034E    75 REM call keyboard data reading subroutine
 0074   034E    80 GOSUB 500
 0077   034E    85 REM if key released code then branch to start of
 0077   034E    86 REM code generation
 0077   034E    90 T=A AND &H80
 0080   0352    100 IF T<>0 THEN 80
 0087   0352    105 REM if shift1 pressed then branch to shift1 porti
                on
 0087   0352    110 IF A=&H2A OR A=&H36 THEN 180
 00A6   0352    115 REM if shift2 pressed then branch to shift2 porti
                on
 00A6   0352    120 IF A=&H38 OR A=&H3A THEN 260
 00C5   0352    125 REM if ctrl pressed then branch to ctrl portion
 00C5   0352    130 IF A=&H1D THEN 340
 00CF   0352    135 REM if num lock pressed then branch to
 00CF   0352    136 REM num lock portion
 00CF   0352    140 IF A=&H45 THEN 420
 00D9   0352    145 REM normal portion
 00D9   0352    147 REM get code from normal group
 00D9   0352    150 N=0
 00DF   0354    155 C=CA(N,A)
 00F4   0356    157 REM call demonstration processing subroutine
 00F4   0356    160 GOSUB 570
 00F7   0356    165 REM branch to start of code generation
 00F7   0356    170 GOTO 80
 00FA   0356    175 REM shift1 portion
 00FA   0356    180 N=1
 0100   0356    185 REM call keyboard data reading subroutine
```

```
                                                            PAGE   2
                                                            10-05-86
                                                            15:45:17
Offset  Data    Source LineIBM Personal Computer BASIC Compiler V1.00

 0100   0356    190 GOSUB 500
 0103   0356    195 REM if shift1 released then branch to
 0103   0356    196 REM start of code generation
 0103   0356    200 IF A=&HAA OR A=&HB6 THEN 80
 0124   0356    205 REM if shift1 still pressed then branch to
 0124   0356    206 REM start of shift1 portion
 0124   0356    210 IF A=&H2A OR A=&H36 THEN 190
 0140   0356    215 REM if previuos pressed key released then
 0140   0356    216 REM branch to start of shift1 portion
 0140   0356    220 T=A AND &H80
 0149   0356    230 IF T<>0 THEN 190
 0150   0356    232 REM get code from shift1 group
 0150   0356    235 C=CA(N,A)
 0165   0356    237 REM call demonstration processing subroutine
 0165   0356    240 GOSUB 570
 0168   0356    245 REM branch to start of shift1 portion
 0168   0356    250 GOTO 190
 016B   0356    255 REM shift2 portion
 016B   0356    260 N=2
 0171   0356    265 REM call keyboard data reading subroutine
 0171   0356    270 GOSUB 500
 0174   0356    275 REM if shift2 released then branch to
 0174   0356    276 REM start of code generation
 0174   0356    280 IF A=&HB8 OR A=&HBA THEN 80
 0195   0356    285 REM if shift2 still pressed then branch to
 0195   0356    286 REM start of shift2 portion
 0195   0356    290 IF A=&H38 OR A=&H3A THEN 270
 01B1   0356    295 REM if previous pressed key released then
 01B1   0356    296 REM branch to start of shift2 portion
 01B1   0356    300 T=A AND &H80
 01BA   0356    310 IF T<>0 THEN 270
 01C1   0356    312 REM get code from shift2 group
 01C1   0356    315 C=CA(N,A)
 01D6   0356    317 REM call demonstration processing subroutine
 01D6   0356    320 GOSUB 570
 01D9   0356    325 REM branch to start of shift2 portion
 01D9   0356    330 GOTO 270
 01DC   0356    335 REM ctrl portion
 01DC   0356    340 N=3
 01E2   0356    345 REM call keyboard data reading subroutine
 01E2   0356    350 GOSUB 500
 01E5   0356    355 REM if ctrl released then branch to
```

```
                                                                  PAGE   3
                                                                  10-05-86
                                                                  15:45:17
Offset  Data    Source LineIBM Personal Computer BASIC Compiler V1.00

 01E5   0356    356 REM start of code generation
 01E5   0356    360 IF A=&H9D THEN 80
 01F0   0356    365 REM if ctrl still pressed then branch to
 01F0   0356    366 REM start of ctrl portion
 01F0   0356    370 IF A=&H1D THEN 350
 01F7   0356    375 REM if previous pressed key released then
 01F7   0356    376 REM branch to start of ctrl portion
 01F7   0356    380 T=A AND &H80
 0200   0356    390 IF T<>0 THEN 350
 0207   0356    392 REM get code from ctrl group
 0207   0356    395 C=CA(N,A)
 021C   0356    397 REM call demonstration processing subroutine
 021C   0356    400 GOSUB 570
 021F   0356    405 REM branch to start of ctrl portion
 021F   0356    410 GOTO 350
 0222   0356    415 REM num lock portion
 0222   0356    420 N=4
 0228   0356    425 REM call keyboard data reading subroutine
 0228   0356    430 GOSUB 500
 022B   0356    435 REM if num lock pressed again then branch to
 022B   0356    436 REM start of code generation
 022B   0356    440 IF A=&H45 THEN 80
 0235   0356    445 REM if num lock released then branch to
 0235   0356    446 REM start of num lock portion
 0235   0356    450 IF A=&HC5 THEN 430
 023D   0356    455 REM  if previous pressed key released then
 023D   0356    456 REM branch to start of num lock portion
 023D   0356    460 T=A AND &H80
 0246   0356    470 IF T<>0 THEN 430
 024D   0356    472 REM get code from num lock group
 024D   0356    475 C=CA(N;A)
 0262   0356    477 REM call demonstration processing subroutine
 0262   0356    480 GOSUB 570
 0265   0356    485 REM branch to start of num lock portion
 0265   0356    490 GOTO 430
 0268   0356    495 REM keyboard data reading subroutine
 0268   0356    497 REM toggle clock
 0268   0356    500 OUT &H61,&HCC
 026F   0356    510 OUT &H61,&H4C
 0276   0356    515 REM check for any interrupt request present
 0276   0356    520 D=INP(&H20)
 027F   0358    530 D=D AND &H2
```

```
                                                                 PAGE    4
                                                                 10-05-86
                                                                 15:45:17
Offset  Data     Source LineIBM Personal Computer BASIC Compiler V1.00

 0288   0358     540 IF D=0 THEN 520
 028F   0358     545 REM read keyboard data and return
 028F   0358     550 A=INP(&H60)
 0298   0358     560  RETURN
 0299   0358     565 REM demonstration processing subruotine -
 0299   0358     566 REM prints code on screen
 0299   0358     570 IF C<16 THEN PRINT "0";
 02AA   0358     580 PRINT HEX$(C);" ";
 02BA   0350     590 RETURN
 02BB   0358     999 END
 02BE   0358     1000 REM code tables
 02BE   0358     1001 DATA &H1B,&H31,&H32,&H33,&H34,&H35,&H36,&H37,&H3
                 8,&H39
 02BF   0358     1002 DATA &H30,&H40,&HBB.&H08.&H00,&H6A,&H45,&H21,&H6
                 0,&H5F
 02C0   0358     1003 DATA &H55.&H59.&H5C.&H68.&H52,&H69.&H5E.&H00,&H0
                 0,&H46
 02C1   0358     1004 DATA &H51,&H50,&H57,&H5B,&H66:&H6B,&H65,&H54,&H4
                 1,&H67
 02C2   0358     1005 DATA &HA5,&H00,&H6C,&H4E,&H4F,&H4D,&H43,&H4B,&H4
                 8,&H5D
 02C3   0358     1006 DATA &H58,&H42,&H6D,&H00,&HB9,&H00,&H20,&H00,&H0
                 0,&H00
 02C4   0358     1007 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                 0,&H00
 02C5   0358     1008 DATA &H00,&H00,&H00,&H40,&H00,&H00,&H00,&HB3,&H0
                 0,&H00
 02C6   0358     1009 DATA &H00,&H00,&HFF
 02C7   0358     1010 DATA &H00,&H3F,&HBD,&HB1,&HB2,&HB8,&HB4,&HBC,&HB
                 9,&HBE
 02C8   0358     1011 DATA &HBF,&HC4,&HB3,&H08,&H00,&H26,&H70,&H00,&H2
                 C,&H7E
 02C9   0358     1012 DATA &H78,&H7A,&H7B,&H24,&H76,&H25,&H7D,&H00,&H0
                 0,&H71
 02CA   0358     1013 DATA &H00,&H64,&H79,&HA0,&H22,&H27,&H29,&H77,&H6
                 E,&H23
 02CB   0358     1014 DATA &H5A,&H00,&H28,&H2E,&H75,&H74,&H6F,&H4C,&H7
                 2,&H7C
 02CC   0358     1015 DATA &H49.&H4A,&H2A,&H00,&H00,&H00,&H20,&H00,&H0
                 0,&H00
 02CD   0358     1016 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                 0,&H00
```

```
                                                                   PAGE   5
                                                                    10-05-86
                                                                    15:45:17
Offset  Data    Source LineIBM Personal Computer BASIC Compiler V1.00

 02CE   0358    1017 DATA &H00,&H00,&H00,&H40,&H00,&H00,&H00,&HB3,&H0
                0,&H00
 02CF   0358    1018 DATA &H00,&H00,&HFF
 02D0   0358    1019 DATA &H00,&HB5,&HB6,&H3A,&H3B,&H3C,&HBA,&HAF,&HB
                0,&HC0
 02D1   0358    1020 DATA &HC1,&HC2,&HC3,&H08,&H00,&H00,&H2B,&H61,&HA
                E,&H00
 02D2   0358    1021 DATA &H00,&HA4,&HA7,&H2D,&H53,&H3D,&HAD,&H00,&H0
                0,&H47
 02D3   0358    1022 DATA &HAC,&HAB,&HA3,&HA6,&H00,&H00,&H00,&HA2,&H0
                0,&HB7
 02D4   0358    1023 DATA &H3E,&H00,&H2F,&H63,&HA1,&H62,&H44,&HAA,&HA
                9,&H00
 02D5   0358    1024 DATA &H56,&HA8,&H73,&H00,&H00,&H00,&H20,&H00,&H0
                0,&H00
 02D6   0358    1025 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                0,&H00
 02D7   0358    1026 DATA &H00,&H00,&H00,&H40,&H00,&H00,&H00,&HB3,&H0
                0,&H00
 02D8   0358    1027 DATA &H00,&H00,&HFF
 02D9   0358    1028 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                0,&H00
 02DA   0358    1029 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                0,&H00
 02DB   0358    1030 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                0,&H00
 02DC   0358    1031 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                0,&H00
 02DD   0358    1032 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                0,&H00
 02DE   0358    1033 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                0,&H00
 02DF   0358    1034 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                0,&H00
 02E0   0358    1035 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                0,&H00
 02E1   0358    1036 DATA &H00,&H00,&H00
 02E2   0358    1037 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                0,&H00
 02E3   0358    1038 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                0,&H00
 02E4   0358    1039 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
```

```
                                                                    PAGE  6
                                                                    10-05-86
                                                                    15:45:17
Offset  Data     Source LineIBM Personal Computer BASIC Compiler V1.00

                 0,&H00
 02E5    0358    1040 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                 0,&H00
 02E6    0358    1041 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                 0,&H00
 02E7    0358    1042 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                 0,&H00
 02E8    0358    1043 DATA &H00,&H00,&H00,&H00,&H00,&H00,&H00,&H00,&H0
                 0,&H00
 02E9    0358    1044 DATA &H37,&H38,&H39,&H40,&H38,&H35,&H36,&HB3,&H3
                 1,&H32
 02EA    0358    1045 DATA &H33,&H30,&HBA
 02EB    0358
 02EE    0358

 22151 Bytes Available
 20504 Bytes Free

     0 Warning Error(s)
     0 Severe  Error(s)
```

```
Start  Stop   Length Name                     Class
00000H 002EDH 002EEH BC_CODE                  CODE
002EEH 00348H 0005BH CODE                     CODE
00350H 0035FH 00010H BC_ICN                   INIT
00360H 00C04H 008A5H BC_IDS                   INIT
00C10H 00E0CH 001FDH INIT                     INIT
00E10H 01A4EH 00C3FH CONST                    RT_DATA
01A50H 01A50H 00000H DATA                     RT_DATA
01A50H 01A50H 00000H COMMON                   BLANK
01A50H 01A50H 00000H CONST                    CONST
01A50H 01A50H 00000H DATA                     DATA
01A50H 01DA7H 00358H BC_DATA                  DATA
01DA8H 01DA8H 00000H BC_FT                    DATA
01DB0H 01DBFH 00010H BC_CN                    DATA
01DC0H 02664H 008A5H BC_DS                    DATA
02670H 0286FH 00200H STACK                    STACK

Origin   Group
00E1:0   DGROUP

Program entry point at 0000:001A
```

machine language subroutines accessible from the main routine for reading BCII code from the code buffer and an interrupt handling routine to generate BCII code from the key-board scan code.

The two assembley language subroutines and the assembley language interrupt handling routine are assembled[17] and linked to produce executable file and the executable file is saved in such a way that the BASIC main routine can be able to access the machine language subroutine.

All modules of this interrupting mode key-board handling routine are discussed seperately in the following articles.

6.6.1 Main Routine

The responsibility of the main routine or the processing routine in the interrupting mode is to get BCII code from the interrupt handler code buffer and to process the code for whatever processing needed by the application program. The main routine can read code in two ways. In the first method, the main routine calls a code reading subroutine which waits until any code is available in the code buffer and then control is transferred back to the calling routine with passing BCII code to the calling routine. In the second method,the main routine calls another code reading subroutine which tests for the availability of code in the code buffer and passes code to the main routine with setting a flag to indicate that a code is passed to the main routine. If no code is available in the code buffer, the subroutine transfers control back to the main routine with setting a flag to indicate that no code is available in the code buffer. In the first method of getting code, the main routine processes the code as needed by the application program. In the second method of getting code,the main routine processes the code as needed by the application program if a code is passed to the main routine, else the main routine do another task as needed by the application program. The processing of the code might be anything needed by the application program, but for demonstration purpose of the present work, the BCII code is printed in the screen. Two probable algorithms of the main routine are given in Fig. 6.4a for two methods of getting codes and the program listing is given in Table 6.3a.

6.6.2 Machine Language Subroutine With BASIC

The two code reading subroutines and the interrupt handling routine are written in assembly language, then assembled and linked to produce machine language subroutine. This machine language subroutine should be accessible from the BASIC main routine and the subroutine should be written such a way that parameters can be passed between the BASIC main routine and the machine language subroutine. Fortunately the BASIC language has set rules[13] of writting such machine language subroutine and of saving the subroutine in such a way that the BASIC program can load the subroutine into memory prior to subroutine call.

The widely used way of getting machine language subroutine code into memory is to load a file, containing the subroutine code saved earlier by the BSAVE command, by the BLOAD command at the outside memory location of the BASIC work area. Methods of creating such a file is described below:

i) The subroutine is written in assembly language, assembled and linked with the /H switch to produce .EXE file such that the file is loaded at the high memory location outside the BASIC work area.

ii) BASIC.COM is loaded under DEBUG[18] and CS,IP,SS,SP,DE,ES register values are recorded using R command where the BASIC.COM is loaded.

iii) The .EXE file is loaded using DEBUG and CS,IP register values are recorded using R command where the subroutine is loaded.

iv) The registers are reset to value where the BASIC. COM was loaded using R command and branched to BASIC entry point using G command.

v) When BASIC prompt, the BASIC application program is loaded and the DEF SEG and the variable name of the CALL statement are edited with the CS and IP register values where the subroutine was loaded. CS value is used for DEF SEG and IP value is used for the variable name of the CALL statement.

vi) The subroutine area is saved by BSAVE command in direct mode in BASIC using CS and IP register value where the subroutine was loaded as starting address and code length from the LINK MAP.

vii) BASIC application program is edited to contain BLOAD command after DEF SEG that sets the proper value of CS for subroutine.

viii) Resulting modified BASIC application program is then saved.

The simplest way of calling a machine language subroutine from BASIC main routine is to use CALL statement. The syntax of the CALL statement is as below:

```
CALL numvar   (variable list)
```

where numvar is a name of a numeric variable whose value is the offset from the segment set by DEF SEG, i.e., the starting point of the subroutine being called, and variable list is a list of variables seperated by commas that are to be passed as argument (not constant).

Execuation of a CALL statement causes the following:

i) Variables location is pushed onto the stack. The location is specified as a two-byte offset into BASIC's data segment.

ii) The return address specified in the CS and offset are pushed onto the stack.

iii) Control is transferred to using address specified in last DEF SEG and offset specified by numvar.

At entry to the subroutine, DS,ES,SS are set to the address of BASIC's data space and CS contains the value specified by latest DEF SEG. The stack pointer indicates a stack that has only 16 bytes (8 words) available for use in the subroutine. Prior to exit from the subroutine, all segment registers and SP must be restored and the return should be inter-segment return (FAR PROC). If interrupts were disabled, they should be enabled prior to return.

Values can be returned to BASIC through the arguments by changing the values of the variables in the argument list. If the argument is a string, the offset for the argument points to the three- byte string descriptor where the byte 0 indicates the length of the string (0-255), byte 1 indicates low byte of offset of string and byte 2 indicates high byte of offset of string. Parameters are refferenced by adding a positive offset to BP after the called routine moves the current stack pointer into BP. The first instruction in the subroutine should be

```
        PUSH   BP
        MOV    BP,SP
```

The offset into the stack of any one particular argument is calculated as

offset from BP = 2 *(n-m) + 6

where n = total no. of argument passed

m = position of the specific argument.

The return from the subroutine must be with a RET n instruction where n is 2 times the number of arguments.

6.6.3 Code Reading Subroutines

The first code reading subroutine, at the entry to the subroutine, checks if the interrupt vector of the key-board interrupt (type 9) is set or not. If the interrupt vector is not set, it moves the offset address of the interrupt handling routine to the 0024H location and segment address to the 0026H location which is the vector index of the type 9 interrupt. The subroutine then puts the interrupt vector set flag on to indicate that the interrupt vector is set and then checks the code counter for the availability of code in the code buffer. If, at the entry to the subroutine, it finds that the interrupt vector is already set, the control is then transferred to check the code counter. If the code counter is found to be zero, i.e., no code is in the code buffer, the subroutine waits until any code is available in the code buffer. If any code is present, i.e., the code counter is found to be non-zero, the subroutine then gets the BCII code pointed to by the read pointer from the code buffer and passes to the calling routine. It then decrements the code counter by 1 to indicate that one code is read from the code buffer and increments the read pointer by 1 to point the next code to be read and checks whether the read pointer becomes 16 or not after incrementing the pointer. If it becomes 16, it is set to zero and then control is transferred back to the calling routine.

The second code reading subroutine do the same job as the first code reading subroutine but with the exception that after checking the code counter, if the code counter is found to be zero, the control is transferred back to the main routine, rather than waiting for code availability, with setting the status flag to zero to indicate that no code is found in the code buffer. Else, it passes the code pointed to by the read pointer to the main routine with the flag set to 1 to indicate that a code is passed to the main routine and then returns from the subroutine.

Algorithms of two code reading subroutines are given in Fig. 6.4b and the program listing is given in Table 6.3b.

6.6.4 Interrupt Handling Routine

The interrupt handling routine reads the key-board scan code, in response to the key-board interrupt, from the output data register of the interface hardware and generates the BCII code corresponding to the pressed key. It then saves the BCII code into a 16 byte code buffer for later use by the main processing routine. For managing the code buffer, one read pointer which index the buffer location from where code to be read by the code reading subroutine, one write pointer which index the buffer location where the next code to be saved by the interrupt handling routine and a code counter which indicates the number of codes present in the code buffer are used. At the starting of the process, two pointers and the code counter are set to zero. When a code is saved into the buffer, the code counter is incremented by 1 and the write pointer is also incremented by 1 to index the buffer location where the next code to be saved. When a code is read from the buffer, the code counter is decremented by 1 and the read pointer is incremented by 1 to index the buffer location from where the next code to be read. When either of the pointer value becomes 16, it is reset to zero because the buffer length is 16 bytes. The zero code counter value indicates that no code is present in the code buffer and the 16 code counter value indicates that the code buffer is full, i.e., there is no room for a new code to be saved into the buffer unless one is read from the buffer.

Another pointer called the group pointer (GPT) is used which indicates the group of code table from where the BCII code to be generated corresponding to any key-board scan code. This pointer is normally set to zero and is set to 1,2,3 and 4 when either of the shift 1 keys, either of the shift 2 key, the ctrl key and the Num Lock key is pressed respectively.

The interrupt handling routine (INTR) first sets the interrupt enable flag so that any other interrupt request can occur and then saves the previous register values. It then reads the key-board scan code from the data output register of the interface hadware (port 60H) and toggles the clock in the 7th bit of the port 61H. The routine then tests the code counter value for 16. If the code counter value is found to be 16, it calls the BFULL procedure and control is transferred to be prepared for returning from the interrupt routine. Else, the routine tests the group pointer value for 0, 1,2,3 and 4 and calls one of the CODE0, CODE1,CODE2,CODE3 and CODE4 procedures corresponding to the group pointer value found. It then gets prepared for returning from the interrupt routine. Prior to return from the interrupt routine, it clears the interrupt flag so that no interrupt can occur before returning from the interrupt routine. It then sends the non-specific end of interrupt command to the Operation Control Word 2(OCW2) (port 20H) of the 8259 PIC A 20H in this OCW2 commands the non-specific end of interrupt. It then restores all saved registers and returns to the interrupted main routine. During returning from the interrupt routine, the CPU POPes the status flags from the stack, which was saved during entering the interrupt routine, restoring the interrupt enable flag set though the same is cleared within the interrupt routine.

The CODE0 procedure is called by the interrupt handling routine which first tests whether the previously pressed key is released or not. If the previously pressed key is released, the procedure returns to the calling routine. Else, the procedure tests for any one of the shift 1 keys, shift 2 keys, ctrl key and Num Lock key pressed. If any one of these keys is pressed, the procedure sets the corresponding group pointer value and returns to the calling routine. Else, it generates the BCII code corresponding to the normal symbol assigned to the pressed key from the normal group of the code table. The procedure then saves the generated BCII code into the code buffer at the location pointed to by the write pointer. It then calls the PSET procedure and returns to the calling routine.

The CODE1 procedure is called by the interrupt handling routine which first tests whether the shift 1 key is released or not. If the shift 1 key is released, it sets the group pointer value to zero and returns to the calling routine. Else, the procedure tests whether the shift 1 key is still pressed or not. If the shift 1 key is still pressed, it returns to the calling routine. Else, it tests whether the previously pressed key is released or not. If the previously pressed key is released, the procedure returns to the calling routine. Else, it generates the BCII code corresponding to the symbol assigned to the shift 1 position of the pressed key from the shift 1 group of the code table. The procedure then saves the generated BCII code into the code buffer at the location pointed to by the write pointer. It then calls the PSET procedure and returns to the calling routine.

The CODE2 and CODE3 procedures are called by the interrupt handling routine and do the similar tasks as that of the CODE1 procedure.

The CODE4 procedure is called by the interrupt handling routine which first tests whether the Num Lock key is pressed again or not, because the Num Lock key is toggle in nature. If the Num Lock key is pressed again, the procedure set the group pointer value to zero and returns to the calling routine. Else, it tests whether the Num Lock key is released or not. If the Num Lock key is released, the procedure returns to the calling routine. Else, it tests whether the previously pressed key is released or not. If the previously pressed key is released, the procedure returns to the calling routine. Else, it generates the BCII code corresponding to the numeral assigned to the pressed keypad key from the Num Lock group of the code table. The procedure then saves the generated BCII code into the code buffer at the location pointed to by the write pointer. It then calls the PSET procedure and returns to the calling routine.

The BFULL procedure is called by the interrupt handling routine which tests the group pointer value for 0,1,2,3 and 4 and calls one of the FULL0,FULL1 ,FULL2 ,FULL3 and FULL4 procedures corresponding to the group pointer value found. It then calls the BEEP procedure and returns to the calling routine.

The FULL0 procedure is called by the BFULL procedure which first tests whether the previously pressed key is released or not. If the previously pressed key is released, the procedure returns to the calling procedure. Else, it tests whether any one of the shift 1, shift 2, ctrl and Num Lock keys is pressed or not. If any one of these keys is pressed, the procedure sets the corresponding group pointer value and returns to the calling procedure.

The FULL1 , FULL2 , FULL3 and FULL4 procedures are called by the BFULL procedure. The FULL1 ,FULL2 and FULL3 procedures tests whether the shift 1, shift 2 and ctrl key respectively is released or not. If the corresponding key is released, these procedures set the group pointer value to zero and return to the calling procedure. The FULL4 procedure tests whether the Num Lock key is pressed again or not. If the Num Lock key is pressed again, the procedure sets the group pointer value to zero and returns to the calling procedure.

Actually the FULL0, FULL1 , FULL2 , FULL3 and FULL4 procedures do the similar job as that of the CODE0,CODE1 ,CODE2 , CODE3 and CODE4 procedures respectively except that these procedures do not generate any BCII code. These procedures are called only when the code buffer is full to keep track of the group pointer value.

The PSET procedure is called by the CODE0,CODE1 ,CODE2 , CODE3 and CODE4 procedures. This procedure first increments the write pointer value by 1 to index the location of the code buffer where the next code to be saved and tests whether the write pointer value becomes 16 or not. If the write pointer value becomes 16, it sets the write pointer value to zero. The procedure then increments the code counter value by 1 and tests whether the code counter value becomes 16 or not. If the code counter value becomes 16 indicating that the code buffer gets full, the procedure calls the BEEP procedure and then returns to the calling procedure.

The BEEP procedure is called by the interrupt handling routine and the PSET procedure. The purpose of this procedure is to give beep to inform the user that the code buffer is full and the next code will not be saved unless any code is read from the code buffer. This procedure first enables the speaker and turns on the modulating signal to the speaker by writting logic 1 in bit 0 and bit 1 of the port 61H respectively. The procedure then gives a delay for a time for which the speaker continues to sound and then the speaker is disabled and the modulating signal to the speaker is turned off by writting logic 0 in the corresponding bits of the port 61H. The procedure then returns to the calling routine.

The algorithm of the interrupt handling routine is given in Fig. 6.4c and the program listing is given in Table 6.3b.

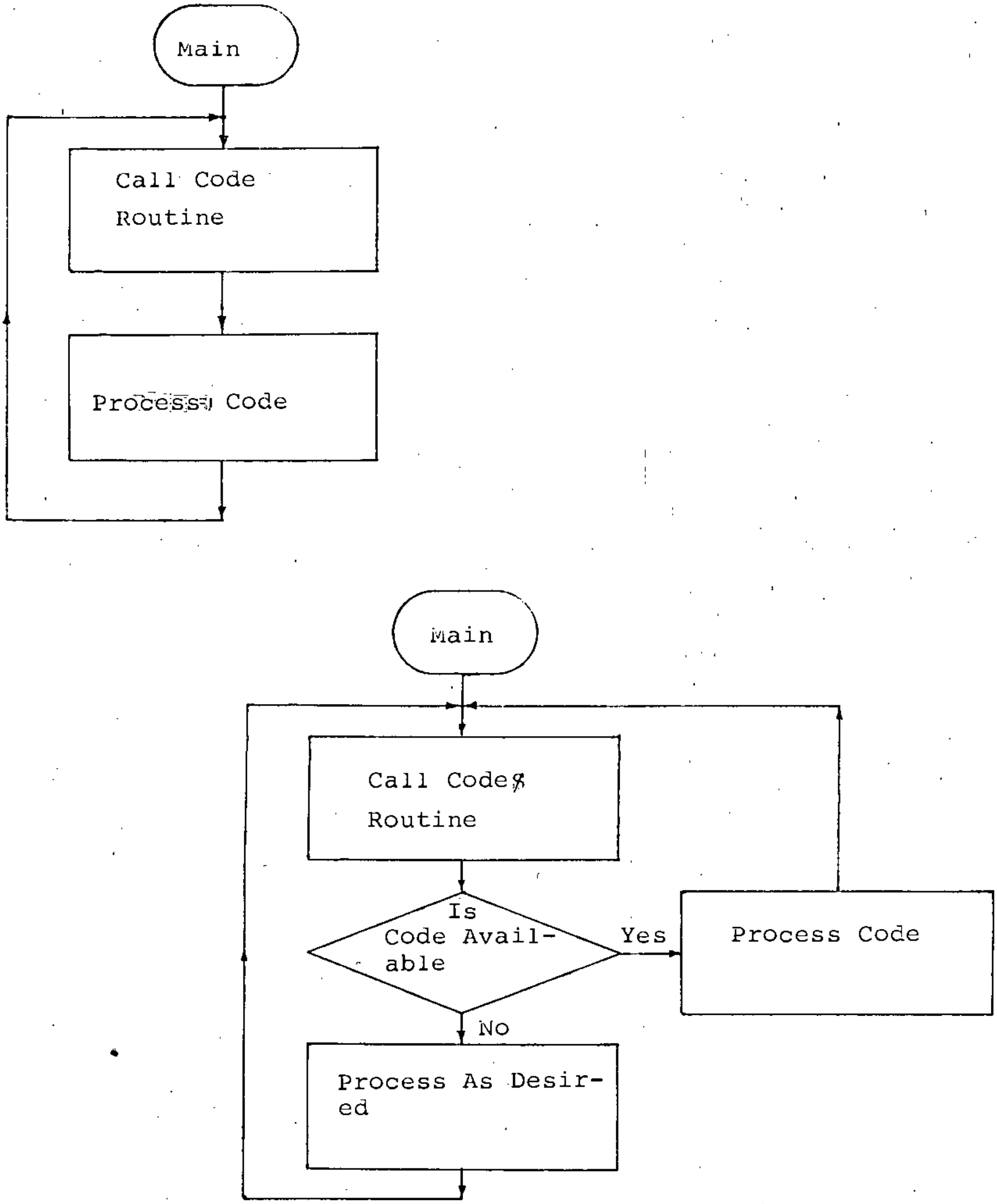


Fig. 6.4a: Algorithm of Main Routine in Interrupting Mode.

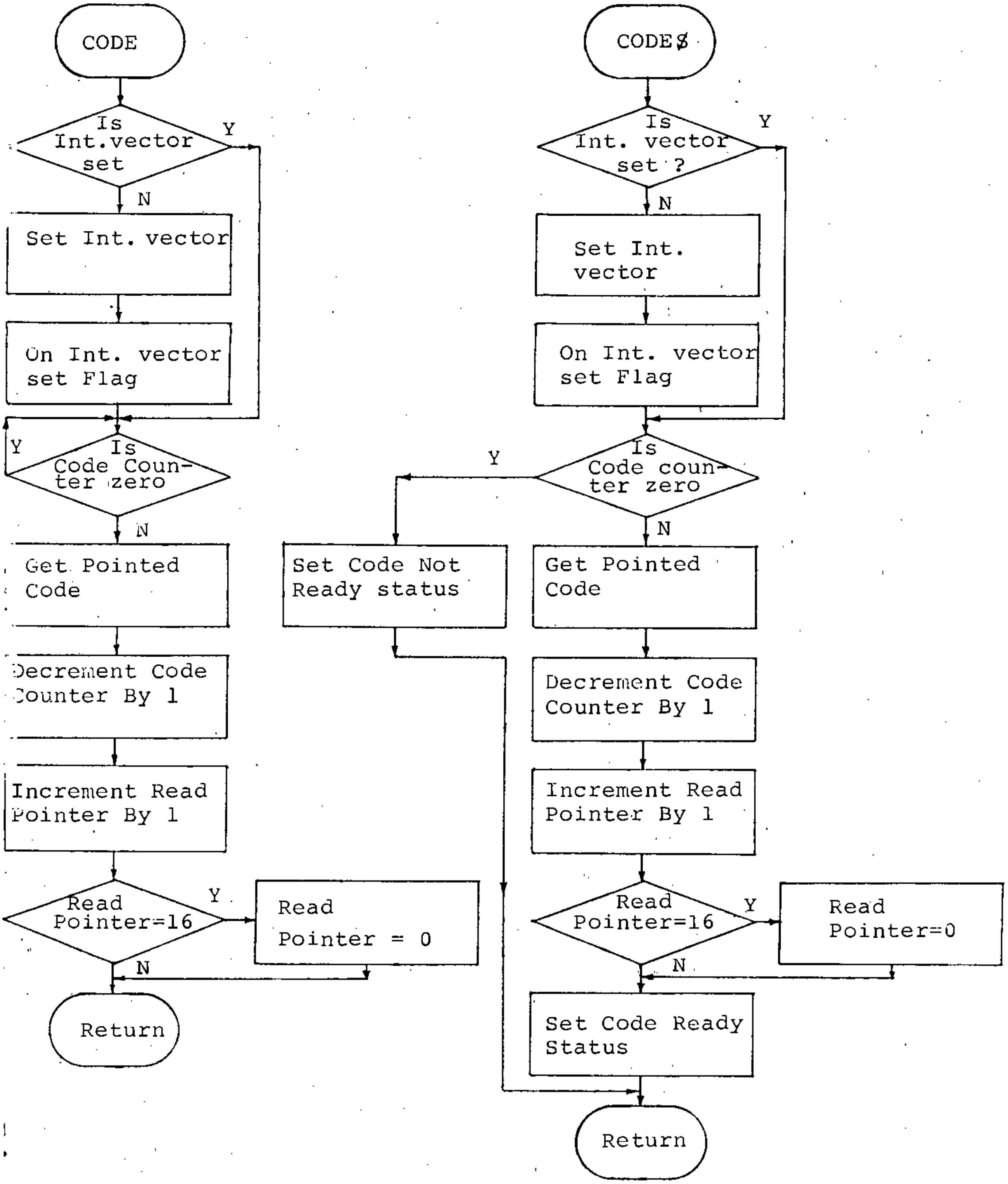


Fig. 6.4b: Algorithm of Code Reading Subroutines in Interrupting Mode.

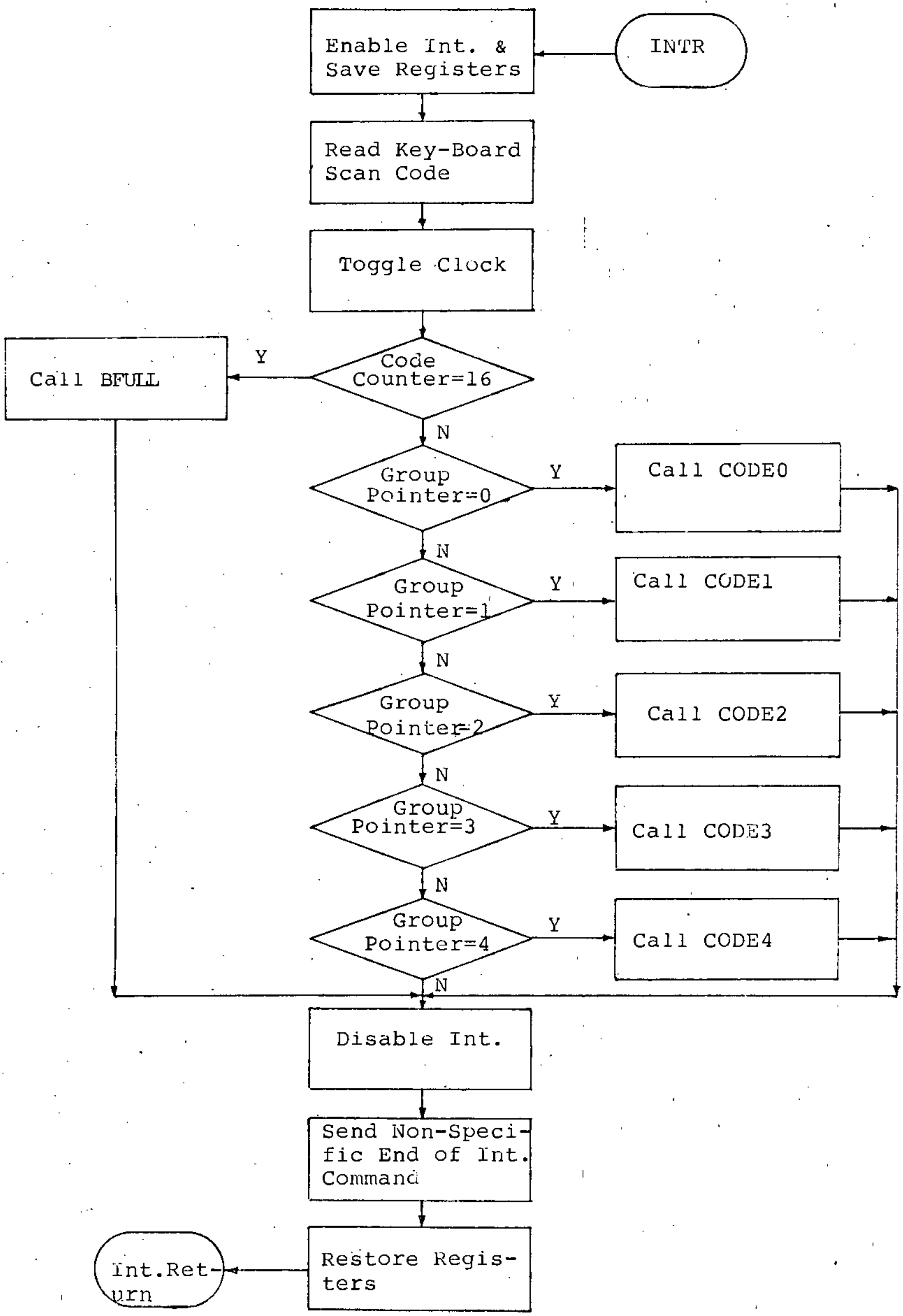


Fig. 6.4c: Algorithm of Interrupt Handling Routine in Interrupting Mode.

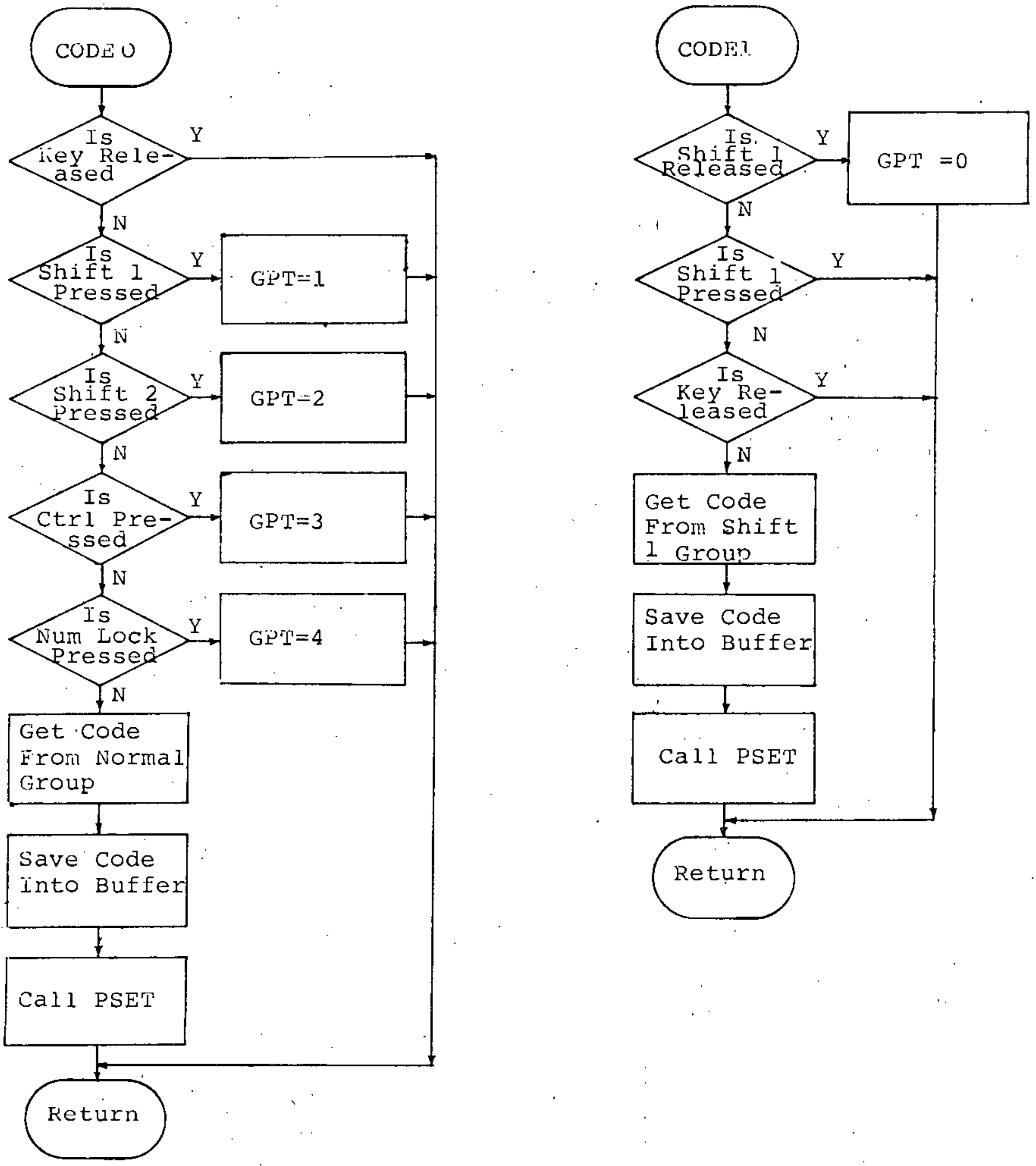


Fig. 6.4c Contd.

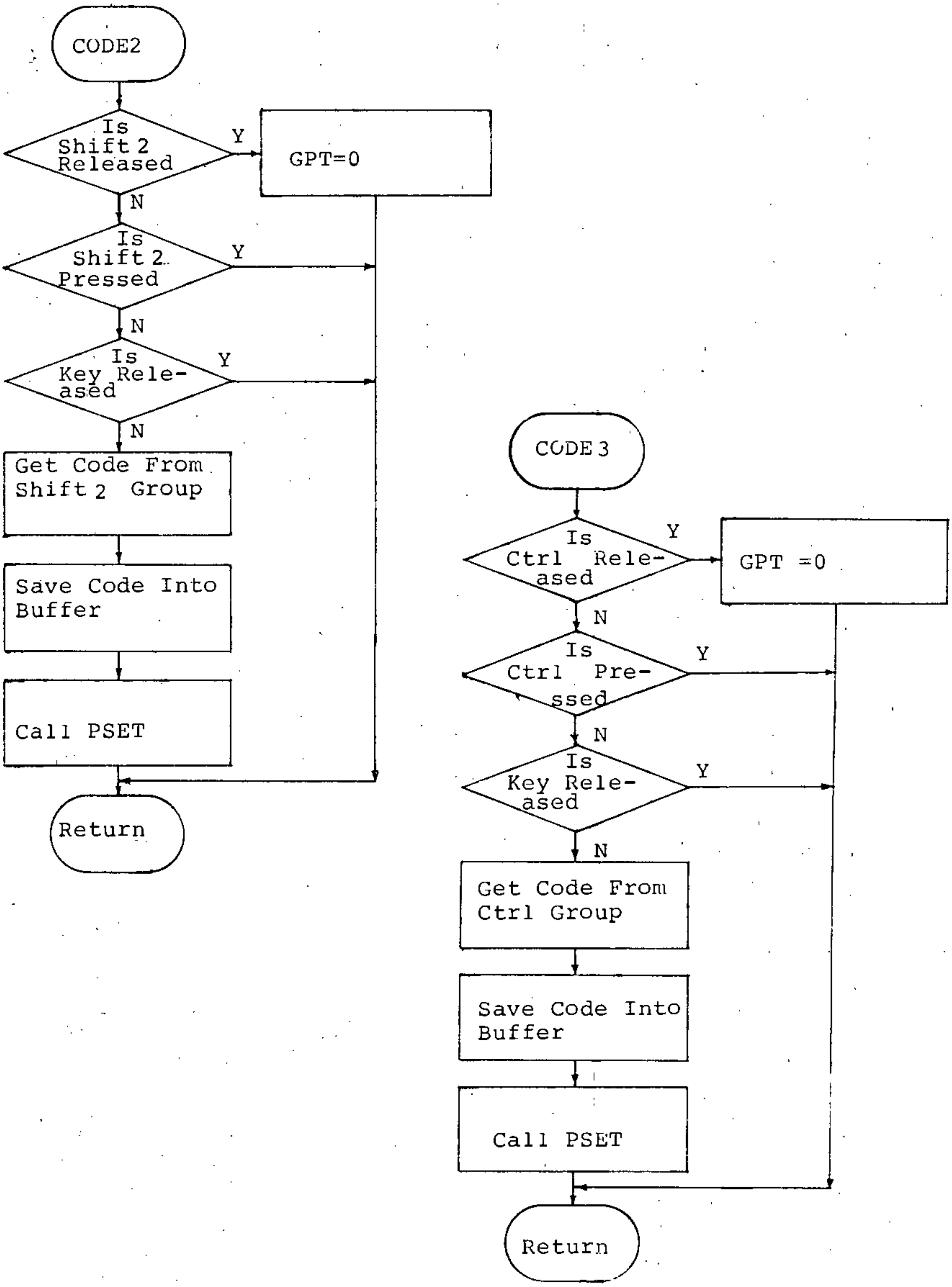


Fig. 6.4c Contd.

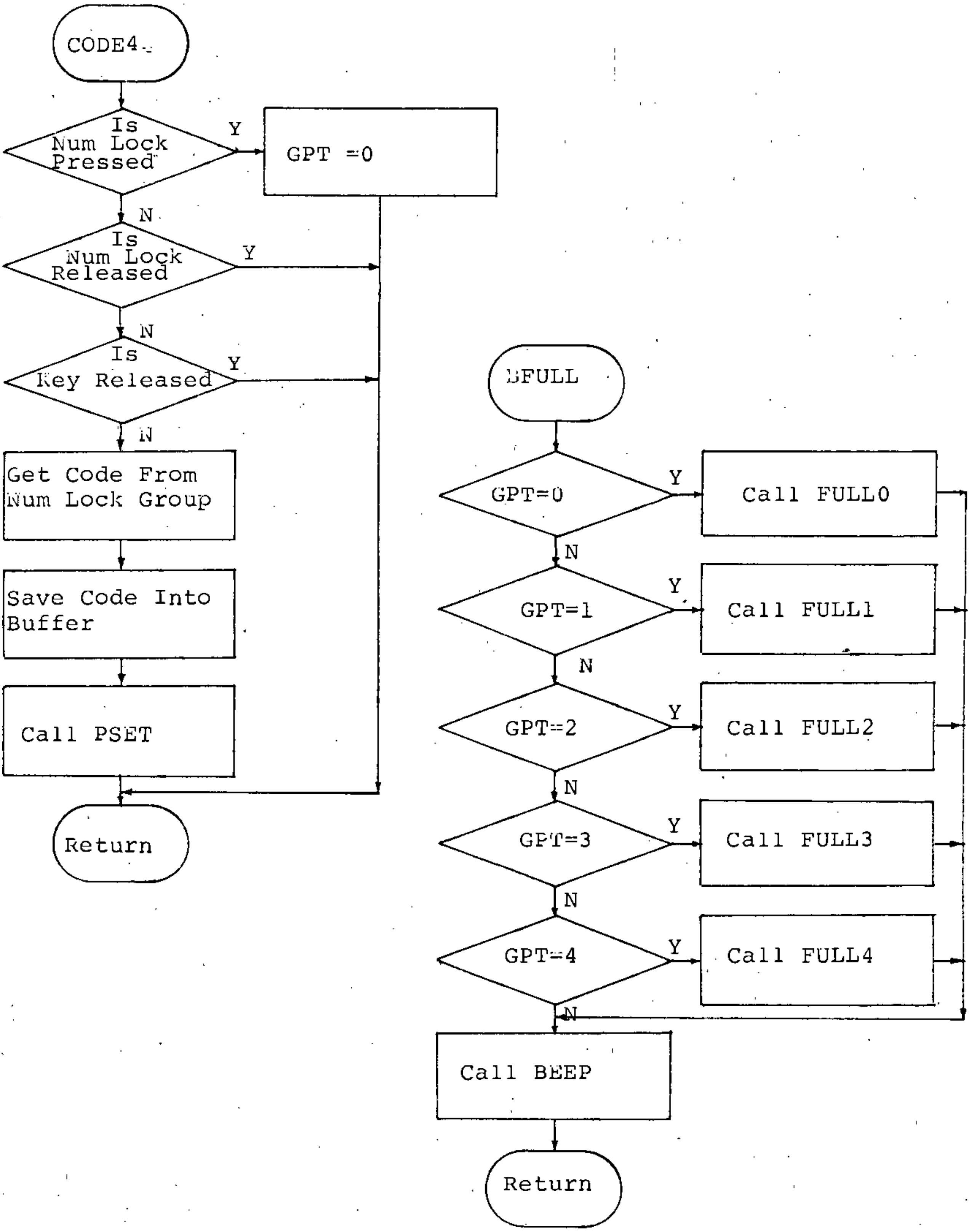


Fig. 6.4c Contd.

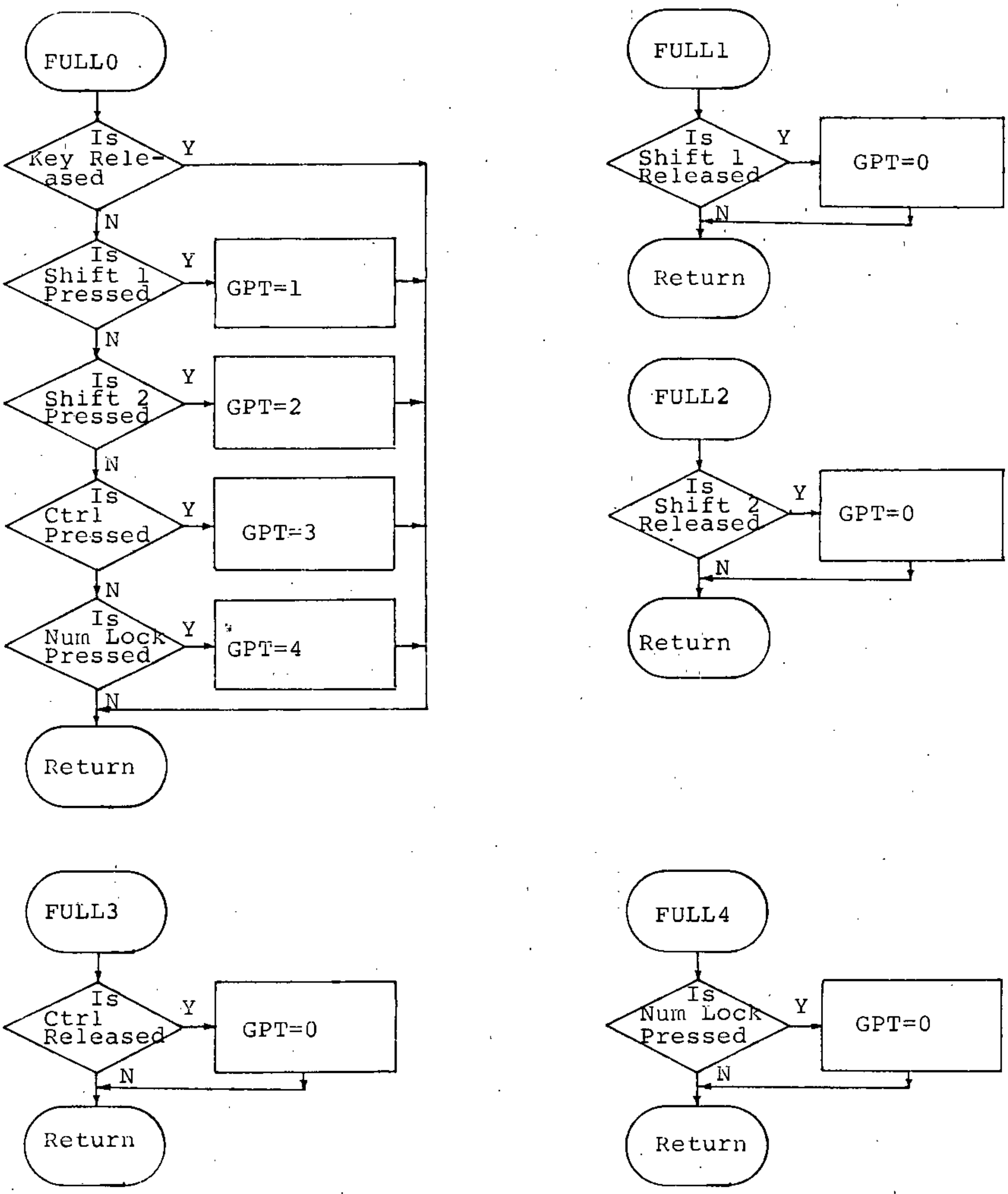


Fig. 6.4c Contd.

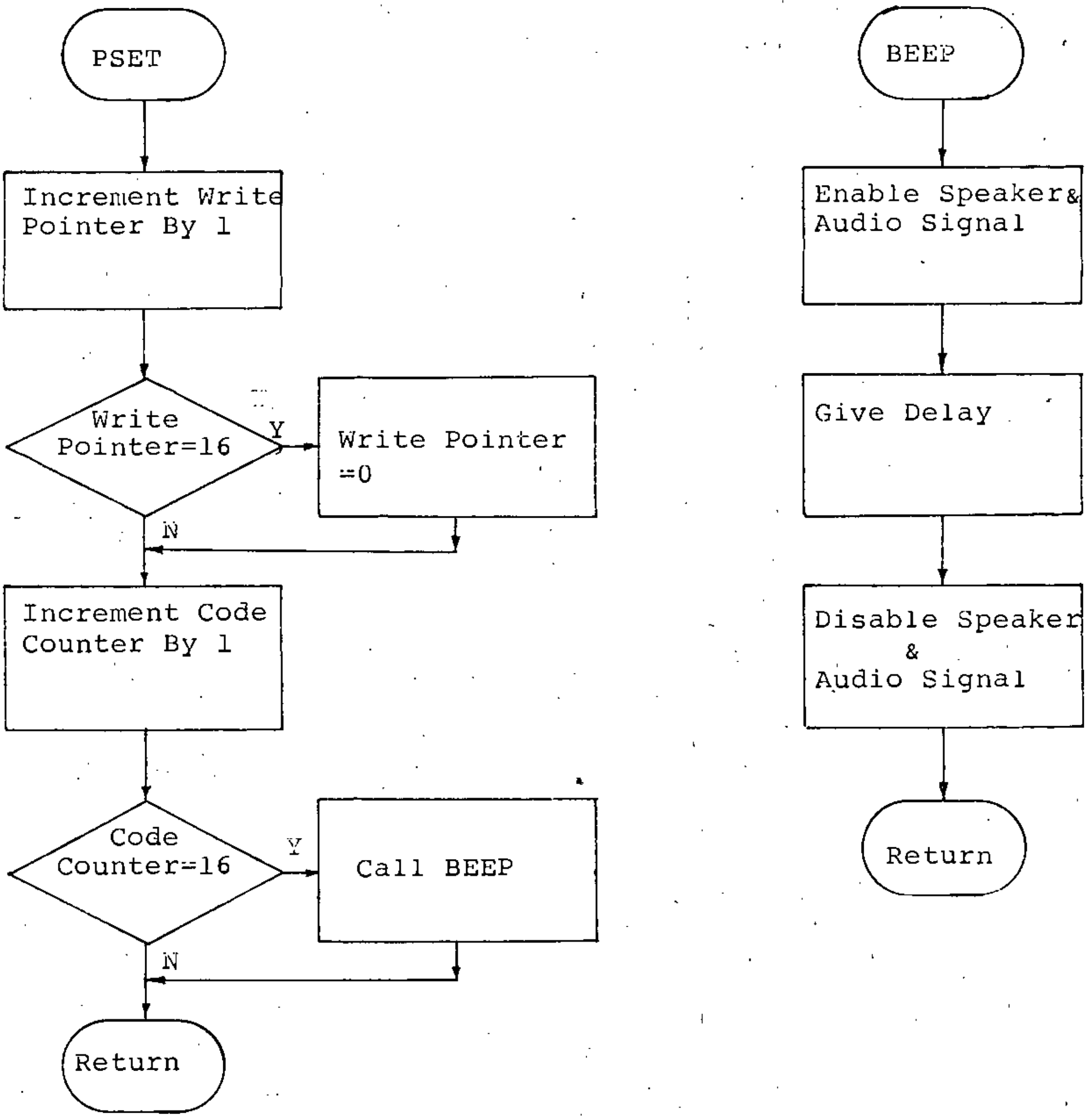


Fig. 6.4c Contd.

Table 6.3a: Program Listing of Main Routine of the Interrupting Mode Key-Board Handling Routine.

```
5  REM set seg addr of subrtn
10 DEF SEG=&H3FA0
15 REM load subrtn into memory
20 BLOAD "SUBRT.EXE",0
25 REM call subrtn
30 V=0
40 CALL V(CODE%)
95  REM print BCII code on screen
100 IF CODE%<16 THEN PRINT "0";
110 PRINT HEX$(CODE%);" ";
115 REM call subrtn again
120 GOTO 40
130 END
```

```
5  REM set seg addr of subrtn
10 DEF SEG=&H3FA0
15 REM load subrtn into memory
20 BLOAD "SUBRT.EXE",0
25 REM call subrtn
30 V1=&H66
40 CALL V1(C%,ST%)
45 REM if no code fuond,call again
50  IF ST%=0 THEN 40
55 REM if code found,print on screen
60 IF C%<16 THEN PRINT "0";
70 PRINT  HEX$(C%);" ";
75 REM call subrtn again
80 GOTO 40
90 END
```

Table 6.3b: Program Listing of Code Reading Subroutines And Interrupt Handling Routine Of The Interrupting Mode Key-Board Handling Routine.

```
The Microsoft MACRO Assembler , Version 1.27          Page    1-

                                 1
                                                             10-05-86


                                      PAGE     40,75
                                CGROUP GROUP    CSEG,DSEG
0000                            CSEG   SEGMENT
                                      ASSUME   CS:CGROUP,DS:CGROUP,ES:CGR
                                OUP
                                ;code reading subruotine which waits until
                                l
                                ;any code available
0000                            CODE   PROC     FAR
0000  55                              PUSH     BP        ;save BP & load wi
                                th SP
0001  8B EC                           MOV      BP,SP     ;for getting arg.
                                addr.
0003  06                              PUSH     ES        ;save ES & use for

                                                         ;addressing sub. d
                                ata
0004  B8 ---- R                       MOV      AX,CGROUP ;load ES with gr
                                oup
0007  8E C0                           MOV      ES,AX       ;seg addr
0009  26: 80 3E 0000 R FF             CMP      BYTE PTR ES:F,0FFH ;if int
                                .
000F  74 1E                           JE       NT  ;vec. set,read code
0011  1E                              PUSH     DS  ;save DS for use in su
                                b.
0012  B8 0000                         MOV      AX,0000H  ;load DS with se
                                g
0015  8E D8                           MOV      DS,AX       ;of int vec tabl
                                e
0017  B8 00DE R                       MOV      AX,OFFSET CGROUP:INTR ;set
001A  BF 0024                         MOV      DI,0024H  ;int vec of key
001D  89 05                           MOV      WORD PTR [DI],AX ;boadr
001F  B8 ---- R                       MOV      AX,SEG CGROUP:INTR ;int by
```

```
The Microsoft MACRO Assembler , Version 1.27            Page    1-
                                          2
                                                                   10-05-86


 0022  BF 0026                             MOV     DI,0026H ;addr of int
 0025  89 05                               MOV     WORD PTR [DI],AX ;routine
 0027  1F                                  POP     DS        ;restore DS
 0028  26: C6 06 0000 R FF                 MOV     ES:F,0FFH ;on int set flag
       90
 002F  26: 80 3E 0013 R 00         NT:     CMP     ES:CTR,00H ;check for code
 0035  74 F8                               JE      NT         ;availability
 0037  26: 8B 36 0015 R                    MOV     SI,ES:RPT  ;load pointed
 003C  B4 00                               MOV     AH,0H      ;code in AL wit
                                   h AH
 003E  26: 8A 84 0001 R                    MOV     AL,BYTE PTR ES:BUFF[SI] ;=
                                   0
 0043  8B 7E 06                            MOV     DI,[BP]+6  ;pass code in
 0046  89 05                               MOV     [DI],AX    ;calling routin
                                   e
 0048  26: FE 0E 0013 R                    DEC     ES:CTR     ;update code co
                                   unt
 004D  26: FF 06 0015 R                    INC     ES:RPT     ;update
 0052  26: 83 3E 0015 R 10                 CMP     ES:RPT,16  ;read
 0058  75 07                               JNE     R          ;
 005A  26: C7 06 0015 R 0000               MOV     ES:RPT,0H  ;pointer
 0061  07                          R:      POP     ES         ;save ES
 0062  5D                                  POP     BP         ;& BP
 0063  CA 0002                             RET     2          ;return
 0066                              CODE    ENDP
                                   ;code reading subroutine which pass code
                                   ;with flag setting
 0066                              CODE$   PROC    FAR
 0066  55                                  PUSH    BP     ;save BP &
 0067  8B EC                               MOV     BP,SP ;load with SP
 0069  06                                  PUSH    ES     ;save ES for use
 006A  B8 ---- R                           MOV     AX,CGROUP ;load ES with
 006D  8E C0                               MOV     ES,AX     ;seg of group
```

```
The Microsoft MACRO Assembler , Version 1.27          Page     1-
                                                3
                                                                    10-05-86


 006F   26: 80 3E 0000 R FF                    CMP      BYTE PTR ES:F,0FFH ;if int
 0075   74 1D                                  JE       NT$ ;vec set,read code
 0077   1E                                     PUSH     DS        ;save DS
 0078   B8 0000                                MOV      AX,0000H ;load DS with seg
 007B   8E D8                                  MOV      DS,AX    ;addr of int tabl
                                        e
 007D   B8 00DE R                              MOV      AX,OFFSET CGROUP:INTR ;set
                                         int
 0080   BF 0024                                MOV      DI,0024H       ;vec of key

 0083   89 05                                  MOV      WORD PTR [DI],AX ;board in
                                        t
 0085   B8 ---- R                              MOV      AX,SEG CGROUP:INTR ;by add
                                        r
 0088   BF 0026                                MOV      DI,0026H            ;of int
 008B   89 05                                  MOV      WORD PTR [DI],AX    ;routin
                                        e
 008D   1F                                     POP      DS  ;restore DS
 008E   26: C6 06 0000 R FF                    MOV      BYTE PTR ES:F,0FFH ;on int
                                                                            ;set fl
                                        ag
 0094   26: 80 3E 0013 R 00             NT$:   CMP      ES:CTR,00H ;if no code ,re
                                        t.
 009A   74 35                                  JE       F$            ;with flag set
 009C   26: 8B 36 0015 R                       MOV      SI,ES:RPT     ;load pointed
 00A1   B4 00                                  MOV      AH,0H         ;code in AL wit
                                        h AH
 00A3   26: 8A 84 0001 R                       MOV      AL,BYTE PTR ES:BUFF[SI] ;=
                                        0
 00A8   8B 7E 08                               MOV      DI,[BP]+8     ;pass code to
 00AB   89 05                                  MOV      [DI],AX       ;calling routin
                                        e
 00AD   26: FE 0E 0013 R                       DEC      ES:CTR    ;update code coun
```

```
The Microsoft MACRO Assembler , Version 1.27           Page    1-
                                4
                                                                10-05-86


                                t
 00B2  26: FF 06 0015 R                INC      ES:RPT    ;update
 00B7  26: 83 3E 0015 R 10             CMP      ES:RPT,16 ;read
 00BD  75 07                           JNE      N$        ;
 00BF  26: C7 06 0015 R 0000           MOV      ES:RPT,0H ;pointer
 00C6  8B 7E 06                 N$:    MOV      DI,[BP]+6 ;pass to calling

 00C9  B8 0001                         MOV      AX,1H     ;rootine code
 00CC  89 05                           MOV      [DI],AX   ;passing flag se
                                t
 00CE  EB 09 90                        JMP      R$
 00D1  8B 7E 06                 F$:    MOV      DI,[BP]+6 ;pass to calling
 00D4  B8 0000                         MOV      AX,0H     ;routine code no
                                t
 00D7  89 05                           MOV      [DI],AX   ;passing flag se
                                t
 00D9  07                       R$:    POP      ES  ;restore
 00DA  5D                              POP      BP  ;ES & BP
 00DB  CA 0004                         RET      4   ;return
 00DE                           CODE$  ENDP
                                ;interrupt handling routine
 00DE                           INTR   PROC     NEAR
 00DE  FB                              STI         ;enable int.
 00DF  50                              PUSH     AX ;save
 00E0  53                              PUSH     BX ;
 00E1  51                              PUSH     CX ;
 00E2  52                              PUSH     DX ;
 00E3  56                              PUSH     SI ;
 00E4  57                              PUSH     DI ;
 00E5  1E                              PUSH     DS ;
 00E6  06                              PUSH     ES ;registers
 00E7  E4 60                           IN       AL,60H ;read keyboard scan
 00E9  50                              PUSH     AX     ;code and save
```

```
The Microsoft MACRO Assembler , Version 1.27           Page     1-
                                            5
                                                                10-05-86

00EA  E4 61                            IN      AL,61H ;toggle
00EC  8A E0                            MOV     AH,AL  ;clock
00EE  0C 80                            OR      AL,80H ;at the
00F0  E6 61                            OUT     61H,AL ;7th bit
00F2  86 C4                            XCHG    AL,AH  ;of port
00F4  E6 61                            OUT     61H,AL ;60H
00F6  B8 ---- R                        MOV     AX,CGROUP ;load DS by seg
00F9  8E D8                            MOV     DS,AX     ;addr of group
00FB  58                               POP     AX ;restore kb scan code
00FC  80 3E 0013 R 10                  CMP     CTR,16 ;if buff not full,
0101  75 06                            JNE     N1     ;gen. code.else
0103  E8 027C R                        CALL    BFULL  ;call BFULL &
0106  EB 3F 90                         JMP     R1     ;return
0109  80 3E 0014 R 00          N1:     CMP     GPT,00H ;if group pointer
010E  75 06                            JNE     N2      ;0, call
0110  E8 0188 R                        CALL    CODE0   ;CODE0 &
0113  EB 32 90                         JMP     R1      ;return
0116  80 3E 0014 R 01          N2:     CMP     GPT,1H  ;if group pointer
011B  75 06                            JNE     N3      ;1,call
011D  E8 01D8 R                        CALL    CODE1   ;CODE1 &
0120  EB 25 90                         JMP     R1      ;return
0123  80 3E 0014 R 02          N3:     CMP     GPT,2H  ;if group pointer
0128  75 06                            JNE     N4      ;2, call
012A  E8 0205 R                        CALL    CODE2   ;CODE2 &
012D  EB 18 90                         JMP     R1      ;return
0130  80 3E 0014 R 03          N4:     CMP     GPT,3H  ;if group pointer
0135  75 06                            JNE     N5      ;3, call
0137  E8 0232 R                        CALL    CODE3   ;CODE3 &
013A  EB 0B 90                         JMP     R1      ;return
013D  80 3E 0014 R 04          N5:     CMP     GPT,4H  ;if group pointer
0142  75 03                            JNE     R1      ;4, call CODE4
0144  E8 0257 R                        CALL    CODE4   ;& return
0147  FA                       R1:     CLI             ;disable interrupt
```

```
The Microsoft MACRO Assembler , Version 1.27           Page    1-
                                   6
                                                              10-05-86


0148  B0 20                              MOV      AL,20H  ;send non-specific

014A  E6 20                              OUT      20H,AL  ;end of int. comma
                                   nd
014C  07                                 POP      ES      ;restore
014D  1F                                 POP      DS      ;
014E  5F                                 POP      DI      ;
014F  5E                                 POP      SI      ;
0150  5A                                 POP      DX      ;
0151  59                                 POP      CX      ;
0152  5B                                 POP      BX      ;
0153  58                                 POP      AX      ;registers
0154  CF                                 IRET             ;return
0155                               INTR  ENDP
                                   ;gives audio signal on buff full
0155                               BEEP  PROC     NEAR
0155  E4 61                             IN       AL,61H    ;enable
0157  8A E0                             MOV      AH,AL     ;speaker
0159  0C 03                             OR       AL,03H    ;& audio
015B  E6 61                             OUT      61H,AL    ;signal
015D  B9 FFFF                            MOV      CX,0FFFFH ;give
0160  90                           A:    NOP                ;
0161  E2 FD                             LOOP     A         ;delay
0163  8A C4                             MOV      AL,AH  ;disable speaker
0165  E6 61                             OUT      61H,AL ;& audio signal
0167  C3                                RET             ;return
0168                               BEEP  ENDP
                                   ;write pointer & code counter updating
0168                               PSET  PROC     NEAR
0168  FF 06 0011 R                       INC      WPT    ;update
016C  83 3E 0011 R 10                    CMP      WPT,16 ;write pointer
0171  75 06                              JNE      N6     ;after saving
0173  C7 06 0011 R 0000                  MOV      WPT,0H ;one code
```

```
0179  FE 06 0013 R               N6:   INC      CTR     ;update code
017D  80 3E 0013 R 10                  CMP      CTR,16  ;counter and
0182  75 03                            JNE      R2      ;call BEEP if
0184  E8 0155 R                        CALL     BEEP    ;buff full
0187  C3                         R2:   RET              ;return
0188                             PSET  ENDP
                                 ;normal code group
0188                             CODE0 PROC     NEAR
0188  A8 80                            TEST     AL,80H ;if key released,
018A  75 4B                            JNZ      R3      ;return
018C  3C 2A                            CMP      AL,2AH ;if shift1,
018E  74 26                            JE       C1      ;set
0190  3C 36                            CMP      AL,36H ;group pointer
0192  74 22                            JE       C1      ;=0
0194  3C 38                            CMP      AL,38H ;if shift2,
0196  74 27                            JE       C2      ;set
0198  3C 3A                            CMP      AL,3AH ;group pointer
019A  74 23                            JE       C2      ;=0
019C  3C 1D                            CMP      AL,1DH ;if ctrl,set
019E  74 28                            JE       C3      ;group pointer=0
01A0  3C 45                            CMP      AL,45H ;if nom lock,set
01A2  74 2D                            JE       C4      ;group pointer=0
01A4  BB 0017 R                        MOV      BX,OFFSET CGROUP:CODE0D ;g
                                 et
01A7  D7                               XLAT     CODE0D          ;code from

01A8  8B 3E 0011 R                     MOV      DI,WPT      ;normal group a
                                 nd
01AC  88 85 0001 R                     MOV      BYTE PTR BUFF[DI],AL ;save
01B0  E8 0168 R                        CALL     PSET  ;call pointer set
01B3  EB 22 90                         JMP      R3    ;and return
01B6  C6 06 0014 R 01 90         C1:   MOV      GPT,1H ;set gpt=1 on shift
                                 1
```

```
The Microsoft MACRO Assembler , Version 1.27              Page    1-
                                             8
                                                                  10-05-86


01BC  EB 19 90                                JMP      R3      ;and return
01BF  C6 06 0014 R 02 90               C2:    MOV      GPT,2H ;set gpt=2 on shift
                                       2
01C5  EB 10 90                                JMP      R3      ;and return
01C8  C6 06 0014 R 03 90               C3:    MOV      GPT,3H ;set gpt=3 on ctrl
01CE  EB 07 90                                JMP      R3      ;and return
01D1  C6 06 0014 R 04 90               C4:    MOV      GPT,4H ;set gpt=4 on num

01D7  C3                               R3:    RET              ;lock and return
01D8                                   CODE0  ENDP
                                       ;shift1 code group
01D8                                   CODE1  PROC     NEAR
01D8  3C AA                                   CMP      AL,0AAH ;if shift1

01DA  74 22                                   JE       C5       ;released
01DC  3C B6                                   CMP      AL,0B6H ;set group pointer
01DE  74 1E                                   JE       C5       ;=0
01E0  3C 2A                                   CMP      AL,2AH   ;if shift1
01E2  74 20                                   JE       R4       ;still
01E4  3C 36                                   CMP      AL,36H   ;present,
01E6  74 1C                                   JE       R4       ;return
01E8  A8 80                                   TEST     AL,80H   ;if previous press
                                       ed
01EA  75 18                                   JNZ      R4       ;key released,retu
                                       rn
01EC  BB 006B R                               MOV      BX,OFFSET CGROUP:CODE1D ;g
                                       et
01EF  D7                                      XLAT     CODE1D   ;code from
01F0  8B 3E 0011 R                            MOV      DI,WPT   ;shiet1 group and
01F4  88 85 0001 R                            MOV      BYTE PTR BUFF[DI],AL ;save
01F8  E8 0168 R                               CALL     PSET  ;call pointer set
01FB  EB 07 90                                JMP      R4     ;and return
01FE  C6 06 0014 R 00 90               C5:    MOV      GPT,0H ;set gpt=0 on shift
```

```
The Microsoft MACRO Assembler , Version 1.27           Page     1-
                                 9
                                                                  10-05-86

                                 1
0204  C3                         R4:    RET                ;released and retur
                                 n
0205                             CODE1  ENDP
                                 ;shift2 code group
0205                             CODE2  PROC       NEAR
0205  3C B8                             CMP        AL,0B8H ;if shift2
0207  74 22                             JE         C6      ;released,
0209  3C BA                             CMP        AL,0BAH ;set
020B  74 1E                             JE         C6      ;group pointer=0
020D  3C 38                             CMP        AL,38H  ;if shift2
020F  74 20                             JE         R5      ;still
0211  3C 3A                             CMP        AL,3AH  ;present,
0213  74 1C                             JE         R5      ;return
0215  A8 80                             TEST       AL,80H ;if priviously pres
                                 sed
0217  75 18                             JNZ        R5      ;key released,retur
                                 n
0219  BB 00BF R                         MOV        BX,OFFSET CGROUP:CODE2D ;g
                                 et
021C  D7                                XLAT       CODE2D  ;code from
021D  8B 3E 0011 R                      MOV        DI,WPT  ;shift2 group and
0221  88 85 0001 R                      MOV        BYTE PTR BUFF[DI],AL ;save
0225  E8 0168 R                         CALL       PSET ;call pointer set
0228  EB 07 90                          JMP        R5    ;and return
022B  C6 06 0014 R 00 90         C6:    MOV        GPT,0H ;set gpt=0 on shift
                                 2
0231  C3                         R5:    RET                ;released and retur
                                 n
0232                             CODE2  ENDP
                                 ;ctrl code group
0232                             CODE3  PROC       NEAR
0232  3C 9D                             CMP        AL,9DH ;if ctrl released,
```

```
0234  74 1A                               JE      C7       ;set gpt=0
0236  3C 1D                               CMP     AL,1DH ;if ctrl still
0238  74 1C                               JE      R6       ;pressed,return
023A  A8 80                               TEST    AL,80H ;if prv key
023C  75 18                               JNZ     R6       ;released,return
023E  BB 0113 R                           MOV     BX,OFFSET CGROUP:CODE3D ;g
                                 et
0241  D7                                  XLAT    CODE3D    ;code from
0242  8B 3E 0011 R                        MOV     DI,WPT    ;ctrl group and
0246  88 85 0001 R                        MOV     BYTE PTR BUFF[DI],AL ;save
024A  E8 0168 R                           CALL    PSET      ;call pointer set
024D  EB 07 90                            JMP     R6        ;and return
0250  C6 06 0014 R 00 90         C7:      MOV     GPT,0H    ;set gpt=0 on ctr
                                 l
0256  C3                         R6:      RET               ;released and ret
                                 urn
0257                             CODE3  ENDP
                                 ;num lock code group
0257                             CODE4  PROC      NEAR
0257  3C 45                               CMP     AL,45H  ;if num lock press
                                 ed
0259  74 1A                               JE      C8       ;again,set gpt=0
025B  3C C5                               CMP     AL,0C5H ;if num lock

025D  74 1C                               JE      R7       ;released,return
025F  A8 80                               TEST    AL,80H  ;if prv pressed ke
                                 y
0261  75 18                               JNZ     R7       ;released,return
0263  BB 0167 R                           MOV     BX,OFFSET CGROUP:CODE4D ;g
                                 et
0266  D7                                  XLAT    CODE4D     ;code from
0267  8B 3E 0011 R                        MOV     DI,WPT     ;num lock group
                                 and
```

```
The Microsoft MACRO Assembler , Version 1.27            Page     1-
                                11
                                                                 10-05-86


 026B  88 85 0001 R                        MOV     BYTE PTR BUFF[DI],AL ;save
 026F  E8 0168 R                           CALL    PSET        ;call pointer se
                                t
 0272  EB 07 90                            JMP     R7          ;and return
 0275  C6 06 0014 R 00 90       C8:        MOV     GPT,0H      ;set gpt=0 on nu
                                m
 027B  C3                       R7:        RET             ;lock pressed & retur
                                n
 027C                           CODE4      ENDP
                                ;code buffer full
 027C                           BFULL      PROC    NEAR
 027C  80 3E 0014 R 00                     CMP     GPT,0H ;if group pointer
 0281  75 06                               JNE     L1     ;=0,
 0283  E8 02BE R                           CALL    FULL0  ;call FULL0
 0286  EB 32 90                            JMP     L5     ;and BEEP
 0289  80 3E 0014 R 01          L1:        CMP     GPT,1H ;if group pointer
 028E  75 06                               JNE     L2     ;=1,
 0290  E8 02FF R                           CALL    FULL1  ;call FULL1
 0293  EB 25 90                            JMP     L5     ;& BEEP
 0296  80 3E 0014 R 02          L2:        CMP     GPT,2H ;if group pointer
 029B  75 06                               JNE     L3     ;=2,
 029D  E8 030E R                           CALL    FULL2  ;call FULL2
 02A0  EB 18 90                            JMP     L5     ;& BEEP
 02A3  80 3E 0014 R 03          L3:        CMP     GPT,3H ;if group pointer
 02A8  75 06                               JNE     L4     ;=3,
 02AA  E8 031D R                           CALL    FULL3  ;call FULL3
 02AD  EB 0B 90                            JMP     L5     ;& BEEP
 02B0  80 3E 0014 R 04          L4:        CMP     GPT,4H ;if group pointer
 02B5  75 03                               JNE     L5     ;=4,
 02B7  E8 0328 R                           CALL    FULL4  ;call FULL4
 02BA  E8 0155 R                L5:        CALL    BEEP   ;& BEEP
 02BD  C3                                  RET            ;return
 02BE                           BFULL      ENDP
```

```
                                  ;buffer full on normal code group
 02BE                             FULL0  PROC     NEAR
 02BE  A8 80                             TEST     AL,80H ;if prv key
 02C0  75 3C                             JNZ      L10    ;released,return
 02C2  3C 2A                             CMP      AL,2AH ;if shift1
 02C4  74 17                             JE       L6     ;pressed,
 02C6  3C 36                             CMP      AL,36H ;set
 02C8  74 13                             JE       L6     ;gpt=2
 02CA  3C 38                             CMP      AL,38H ;if shift2
 02CC  74 18                             JE       L7     ;pressed,
 02CE  3C 3A                             CMP      AL,3AH ;set

 02D0  74 14                             JE       L7     ;gpt=4
 02D2  3C 1D                             CMP      AL,1DH ;if ctrl pressed,
 02D4  74 19                             JE       L8     ;set gpt=3
 02D6  3C 45                             CMP      AL,45H ;if num lock presse
                                  d,
 02D8  74 1E                             JE       L9     ;set gpt=4
 02DA  EB 22 90                          JMP      L10    ;return
 02DD  C6 06 0014 R 01 90         L6:    MOV      GPT,1H ;set gpt=1 on shift
                                  1
 02E3  EB 19 90                          JMP      L10    ;pressed and return
 02E6  C6 06 0014 R 02 90         L7:    MOV      GPT,2H ;set gpt=2 on shift
                                  2
 02EC  EB 10 90                          JMP      L10    ;pressed and return
 02EF  C6 06 0014 R 03 90         L8:    MOV      GPT,3H ;set gpt=3 on ctrl
 02F5  EB 07 90                          JMP      L10    ;pressed and return
 02F8  C6 06 0014 R 04 90         L9:    MOV      GPT,4H ;set gpt=4 on num l
                                  ock
 02FE  C3                         L10:   RET             ;pressed and return
 02FF                             FULL0  ENDP
                                  ;buff full on shift1 code group
 02FF                             FULL1  PROC     NEAR
```

```
02FF  3C AA                          CMP      AL,0AAH ;if
0301  74 04                          JE       L11     ;shift1
0303  3C B6                          CMP      AL,0B6H ;released,
0305  75 06                          JNE      L12     ;set
0307  C6 06 0014 R 00 90       L11:  MOV      GPT,0H  ;gpt=0
030D  C3                       L12:  RET              ;and return
030E                           FULL1 ENDP
                               ;buff full on shift2 code group
030E                           FULL2 PROC     NEAR
030E  3C B8                          CMP      AL,0B8H ;if
0310  74 04                          JE       L13     ;shift2
0312  3C BA                          CMP      AL,0BAH ;released,
0314  75 06                          JNE      L14     ;set
0316  C6 06 0014 R 00 90       L13:  MOV      GPT,0H  ;gpt=0
031C  C3                       L14:  RET              ;and return
031D                           FULL2 ENDP
                               ;buff full on ctrl code group
031D                           FULL3 PROC     NEAR
031D  3C 9D                          CMP      AL,9DH  ;if ctrl
031F  75 06                          JNE      L15     ;released,
0321  C6 06 0014 R 00 90             MOV      GPT,0H  ;set gpt=0
0327  C3                       L15:  RET              ;and return
0328                           FULL3 ENDP
                               ;buff full on num lock code group
0328                           FULL4 PROC     NEAR
0328  3C 45                          CMP      AL,45H  ;if num lock
032A  75 06                          JNE      L16     ;pressed again,
032C  C6 06 0014 R 00 90             MOV      GPT,0H  ;set gpt=0
0332  C3                       L16:  RET              ;and return
0333                           FULL4 ENDP
0333                           CSEG  ENDS
0000                           DSEG  SEGMENT  BYTE
0000  00                       F     DB       0          ;int set flag
```

```
The Microsoft MACRO Assembler , Version 1.27              Page     1-

                                   14
                                                              10-05-86


0001     10 [                      BUFF    DB        16 DUP(0) ;code buffer
               00
                  ]

0011  0000                         WPT     DW        0          ;write pointer
0013  00                           CTR     DB        0          ;code counter
0014  00                           GPT     DB        0          ;group pointer
0015  0000                         RPT     DW        0          ;read pointer
                                   ;normal group code table
0017  00                           CODE0D DB         0
0018  1B 31 32 33 34 35                    DB        01BH,031H,032H,033H,034H,0
                                   35H,036H,037H,038H,039H

      36 37 38 39
0022  30 40 BB 08 00 6A                    DB        030H,040H,0BBH,008H,000H,0
                                   6AH,045H,021H,060H,05FH

      45 21 60 5F
002C  55 59 5C 68 52 69                    DB        055H,059H,05CH,068H,052H,0
                                   69H,05EH,000H,000H,046H

      5E 00 00 46
0036  51 50 57 5B 66 6B                    DB        051H,050H,057H,05BH,066H,0
                                   6BH,065H,054H,041H,067H

      65 54 41 67
0040  A5 00 6C 4E 4F 4D                    DB        0A5H,000H,06CH,04EH,04FH,0
                                   4DH,043H,04BH,048H,05DH

      43 4B 48 5D
004A  58 42 6D 00 B9 00                    DB        058H,042H,06DH,000H,0B9H,0
                                   00H,020H,000H,000H,000H

      20 00 00 00
0054  00 00 00 00 00 00                    DB        000H,000H,000H,000H,000H,0
                                   00H,000H,000H,000H,000H

      00 00 00 00
005E  00 00 00 40 00 00                    DB        000H,000H,000H,040H,000H,0
                                   00H,000H,0B3H,000H,000H
```

```
      00 B3 00 00
0068  00 00 FF                 DB        000H,000H,0FFH
                          ;shift1 group code table
006B  00                  CODE1D DB      0
006C  00 3F BD B1 B2 B8        DB        000H,03FH,0BDH,0B1H,0B2H,0
                          B8H,0B4H,0BCH,0B9H,0BEH

      B4 BC B9 BE
0076  BF C4 B3 08 00 26        DB        0BFH,0C4H,0B3H,008H,000H,0
                          26H,070H,000H,02CH,07EH

      70 00 2C 7E
0080  78 7A 7B 24 76 25        DB        078H,07AH,07BH,024H,076H,0
                          25H,07DH,000H,000H,071H

      7D 00 00 71
008A  00 64 79 A0 22 27        DB        000H,064H,079H,0A0H,022H,0
                          27H,029H,077H,06EH,023H

      29 77 6E 23
0094  5A 00 28 2E 75 74        DB        05AH,000H,028H,02EH,075H,0
                          74H,06FH,04CH,072H,07CH

      6F 4C 72 7C
009E  49 4A 2A 00 00 00        DB        049H,04AH,02AH,000H,000H,0
                          00H,020H,000H,000H,000H

      20 00 00 00
00A8  00 00 00 00 00 00        DB        000H,000H,000H,000H,000H,0
                          00H,000H,000H,000H,000H

      00 00 00 00
00B2  00 00 00 40 00 00        DB        000H,000H,000H,040H,000H,0
                          00H,000H,0B3H,000H,000H

      00 B3 00 00
00BC  00 00 FF                 DB        000H,000H,0FFH
                          ;shift2 group code table
00BF  00                  CODE2D DB      0
00C0  00 B5 B6 3A 3B 3C        DB        000H,0B5H,0B6H,03AH,03BH,0
                          3CH,0BAH,0AFH,0B0H,0C0H
```

```
      BA AF B0 C0
00CA  C1 C2 C3 08 00 00                 DB          0C1H,0C2H,0C3H,008H,000H,0
                              00H,02BH,061H,0AEH,000H
      2B 61 AE 00
00D4  00 A4 A7 2D 53 3D                 DB          000H,0A4H,0A7H,02DH,053H,0
                              3DH,0ADH,000H,000H,047H
      AD 00 00 47
00DE  AC AB A3 A6 00 00                 DB          0ACH,0ABH,0A3H,0A6H,000H,0
                              00H,000H,0A2H,000H,0B7H
      00 A2 00 B7
00E8  3E 00 2F 63 A1 62                 DB          03EH,000H,02FH,063H,0A1H,0
                              62H,044H,0AAH,0A9H,000H
      44 AA A9 00
00F2  56 A8 73 00 00 00                 DB          056H,0A8H,073H,000H,000H,0
                              00H,020H,000H,000H,000H
      20 00 00 00
00FC  00 00 00 00 00 00                 DB          000H,000H,000H,000H,000H,0
                              00H,000H,000H,000H,000H
      00 00 00 00
0106  00 00 00 40 00 00                 DB          000H,000H,000H,040H,000H,0
                              00H,000H,0B3H,000H,000H
      00 B3 00 00
0110  00 00 FF                          DB          000H,000H,0FFH
                              ;ctrl group code table
0113  00                      CODE3D DB         0
0114  00 00 00 00 00 00                 DB          000H,000H,000H,000H,000H,0
                              00H,000H,000H,000H,000H
      00 00 00 00
011E  00 00 00 00 00 00                 DB          000H,000H,000H,000H,000H,0
                              00H,000H,000H,000H,000H
      00 00 00 00
0128  00 00 00 00 00 00                 DB          000H,000H,000H,000H,000H,0
                              00H,000H,000H,000H,000H
```

```
       00 00 00 00
 0132  00 00 00 00 00 00                    DB        000H,000H,000H,000H,000H,0
                                      00H,000H,000H,000H,000H
       00 00 00 00
 013C  00 00 00 00 00 00                    DB        000H,000H,000H,000H,000H,0
                                      00H,000H,000H,000H,000H
       00 00 00 00
 0146  00 00 00 00 00 00                    DB        000H,000H,000H,000H,000H,0
                                      00H,000H,000H,000H,000H
       00 00 00 00
 0150  00 00 00 00 00 00                    DB        000H,000H,000H,000H,000H,0
                                      00H,000H,000H,000H,000H
       00 00 00 00
 015A  00 00 00 00 00 00                    DB        000H,000H,000H,000H,000H,0
                                      00H,000H,000H,000H,000H
       00 00 00 00
 0164  00 00 00                             DB        000H,000H,000H
                                      ;num lock group code table
 0167  00                             CODE4D DB       0
 0168  00 00 00 00 00 00                    DB        000H,000H,000H,000H,000H,0
                                      00H,000H,000H,000H,000H
       00 00 00 00
 0172  00 00 00 00 00 00                    DB        000H,000H,000H,000H,000H,0
                                      00H,000H,000H,000H,000H
       00 00 00 00
 017C  00 00 00 00 00 00                    DB        000H,000H,000H,000H,000H,0
                                      00H,000H,000H,000H,000H
       00 00 00 00
 0186  00 00 00 00 00 00                    DB        000H,000H,000H,000H,000H,0
                                      00H,000H,000H,000H,000H
       00 00 00 00
 0190  00 00 00 00 00 00                    DB        000H,000H,000H,000H,000H,0
                                      00H,000H,000H,000H,000H
```

```
The Microsoft MACRO Assembler , Version 1.27            Page     1-

                                       18
                                                                  10-05-86


      00 00 00 00
 019A  00 00 00 00 00 00                       DB       000H,000H,000H,000H,000H,0
                                       00H,000H,000H,000H,000H
      00 00 00 00
 01A4  00 00 00 00 00 00                       DB       000H,000H,000H,000H,000H,0
                                       00H,000H,000H,000H,000H
      00 00 00 00
 01AE  37 38 39 40 38 35                       DB       037H,038H,039H,040H,038H,0
                                       35H,036H,0B3H,031H,032H
      36 B3 31 32
 01B8  33 30 BA                                DB       033H,030H,0BAH
 01BB                                  DSEG    ENDS
                                               END
```

```
The Microsoft MACRO Assembler , Version 1.27              Page    Sy

                                mbols-1
                                                                10-05-86


Segments and groups:

                N a m e                     Size    align   combine class

CGROUP . . . . . . . . . . . . . .          GROUP
  CSEG . . . . . . . . . . . . . .          0333    PARA    NONE
  DSEG . . . . . . . . . . . . . .          01BB    BYTE    NONE

Symbols:

                N a m e                     Type    Value   Attr

A. . . . . . . . . . . . . . . . .          L NEAR  0160    CSEG
BEEP . . . . . . . . . . . . . . .          N PROC  0155    CSEG    Length =00
                                  13
BFULL. . . . . . . . . . . . . . .          N PROC  027C    CSEG    Length =00
                                  42
BUFF . . . . . . . . . . . . . . .          L BYTE  0001    DSEG    Length =00
                                  10
C1 . . . . . . . . . . . . . . . .          L NEAR  01B6    CSEG
C2 . . . . . . . . . . . . . . . .          L NEAR  01BF    CSEG
C3 . . . . . . . . . . . . . . . .          L NEAR  01C8    CSEG
C4 . . . . . . . . . . . . . . . .          L NEAR  01D1    CSEG
C5 . . . . . . . . . . . . . . . .          L NEAR  01FE    CSEG
C6 . . . . . . . . . . . . . . . .          L NEAR  022B    CSEG
C7 . . . . . . . . . . . . . . . .          L NEAR  0250    CSEG
C8 . . . . . . . . . . . . . . . .          L NEAR  0275    CSEG
CODE . . . . . . . . . . . . . . .          F PROC  0000    CSEG    Length =00
                                  66
CODE1. . . . . . . . . . . . . . .          F PROC  0066    CSEG    Length =00
                                  78
CODE0. . . . . . . . . . . . . . .          N PROC  0188    CSEG    Length =00
                                  50
```

```
The Microsoft MACRO Assembler , Version 1.27            Page     Sy

                                   mbols-2
                                                                 10-05-86


CODE0D . . . . . . . . . . . . . .          L BYTE   0017     DSEG
CODE1. . . . . . . . . . . . . . .          N PROC   01D8     CSEG      Length =00
                                   2D
CODE1D . . . . . . . . . . . . . .          L BYTE   006B     DSEG
CODE2. . . . . . . . . . . . . . .          N PROC   0205     CSEG      Length =00
                                   2D
CODE2D . . . . . . . . . . . . . .          L BYTE   00BF     DSEG
CODE3. . . . . . . . . . . . . . .          N PROC   0232     CSEG      Length =00
                                   25
CODE3D . . . . . . . . . . . . . .          L BYTE   0113     DSEG
CODE4. . . . . . . . . . . . . . .          N PROC   0257     CSEG      Length =00
                                   25
CODE4D . . . . . . . . . . . . . .          L BYTE   0167     DSEG
CTR. . . . . . . . . . . . . . . .          L BYTE   0013     DSEG
F. . . . . . . . . . . . . . . . .          L BYTE   0000     DSEG
F$ . . . . . . . . . . . . . . . .          L NEAR   00D1     CSEG
FULL0. . . . . . . . . . . . . . .          N PROC   02BE     CSEG      Length =00
                                   41
FULL1. . . . . . . . . . . . . . .          N PROC   02FF     CSEG      Length =00
                                   0F
FULL2. . . . . . . . . . . . . . .          N PROC   030E     CSEG      Length =00
                                   0F
FULL3. . . . . . . . . . . . . . .          N PROC   031D     CSEG      Length =00
                                   0B
FULL4. . . . . . . . . . . . . . .          N PROC   0328     CSEG      Length =00
                                   0B
GPT. . . . . . . . . . . . . . . .          L BYTE   0014     DSEG
INTR . . . . . . . . . . . . . . .          N PROC   00DE     CSEG      Length =00
                                   77
L1 . . . . . . . . . . . . . . . .          L NEAR   0289     CSEG
L10. . . . . . . . . . . . . . . .          L NEAR   02FE     CSEG
L11. . . . . . . . . . . . . . . .          L NEAR   0307     CSEG
L12. . . . . . . . . . . . . . . .          L NEAR   030D     CSEG
```

```
The Microsoft MACRO Assembler , Version 1.27          Page     Sy

                           mbols-3
                                                          10-05-86


L13. . . . . . . . . . . . . . .          L NEAR  0316    CSEG
L14. . . . . . . . . . . . . . .          L NEAR  031C    CSEG
L15. . . . . . . . . . . . . . .          L NEAR  0327    CSEG
L16. . . . . . . . . . . . . . .          L NEAR  0332    CSEG
L2 . . . . . . . . . . . . . . .          L NEAR  0296    CSEG
L3 . . . . . . . . . . . . . . .          L NEAR  02A3    CSEG
L4 . . . . . . . . . . . . . . .          L NEAR  02B0    CSEG
L5 . . . . . . . . . . . . . . .          L NEAR  02BA    CSEG
L6 . . . . . . . . . . . . . . .          L NEAR  02DD    CSEG
L7 . . . . . . . . . . . . . . .          L NEAR  02E6    CSEG
L8 . . . . . . . . . . . . . . .          L NEAR  02EF    CSEG
L9 . . . . . . . . . . . . . . .          L NEAR  02F8    CSEG
N$ . . . . . . . . . . . . . . .          L NEAR  00C6    CSEG
N1 . . . . . . . . . . . . . . .          L NEAR  0109    CSEG
N2 . . . . . . . . . . . . . . .          L NEAR  0116    CSEG
N3 . . . . . . . . . . . . . . .          L NEAR  0123    CSEG
N4 . . . . . . . . . . . . . . .          L NEAR  0130    CSEG
N5 . . . . . . . . . . . . . . .          L NEAR  013D    CSEG
N6 . . . . . . . . . . . . . . .          L NEAR  0179    CSEG
NT . . . . . . . . . . . . . . .          L NEAR  002F    CSEG
NT$. . . . . . . . . . . . . . .          L NEAR  0094    CSEG
PSET . . . . . . . . . . . . . .          N PROC  0168    CSEG    Length =00
                                20
R. . . . . . . . . . . . . . . .          L NEAR  0061    CSEG
R$ . . . . . . . . . . . . . . .          L NEAR  00D9    CSEG
R1 . . . . . . . . . . . . . . .          L NEAR  0147    CSEG
R2 . . . . . . . . . . . . . . .          L NEAR  0187    CSEG
R3 . . . . . . . . . . . . . . .          L NEAR  01D7    CSEG
R4 . . . . . . . . . . . . . . .          L NEAR  0204    CSEG
R5 . . . . . . . . . . . . . . .          L NEAR  0231    CSEG
R6 . . . . . . . . . . . . . . .          L NEAR  0256    CSEG
R7 . . . . . . . . . . . . . . .          L NEAR  027B    CSEG
RPT. . . . . . . . . . . . . . .          L WORD  0015    DSEG
```

```
The Microsoft MACRO Assembler , Version 1.27             Page     Sy
                                                  mbols-4
                                                                   10-05-86


WPT. . . . . . . . . . . . . . . .          L WORD  0011    DSEG

Warning Severe
Errors  Errors
0       0
```

```
Loading High
Warning: No STACK segment

 Start  Stop   Length  Name                     Class
 00000H 00332H 0333H   CSEG
 00333H 004EDH 01BBH   DSEG
```

6.6.5 Calling The Code Reading Subroutines

The two code reading subroutines and the interrupt handling routine are written in assembly language as a single source file. The source file then assembled and linked with /H switch to produce executable file such that the executable file is loaded at the upper available location of the memory. The produced executable file then loaded, in a IBM PC with 256 K byte memory, under DEBUG and saved by the BSAVE command with the file name "SUBRT. EXE" such that the file can be loaded by the BASIC main program prior to subroutine call.

From the LINK MAP it is found that the code length of the file is 4EEH byte. Using DEBUG commands, it is found that the file is loaded at the segment address 3FA0H. The offset address of the code reading subroutine, which waits until any code is available in the code buffer, is 0000H and that of the code reading subroutine, which returns with status flag set to indicate whether any code is found or not, is 0066H.

The main routine should contain the following two statements prior to any subroutine call to specify the segment address of the subroutines and to load the subroutine code into memory:

```
DEF SEG = &H3FA0
BLOAD "SUBRT. EXE",0
```

If the main routine calls the code reading subroutine which waits until any code is available in the code buffer, the CALL statement should be as below:

```
numvar = 0

CALL numvar (argu)
```

where the numvar is any numeric variable which is assigned the value of the offset address of the called code reading subroutine. After returning from the subroutine, the argument will contain the BCII code read from the code buffer. If the main routine calls the code reading subroutine which returns with status flag set to indicate whether any code is found or not, the CALL statement should be as below:

```
numvar = &H66

CALL numvar (argu 1, argu 2)
```

where the numvar is any numeric variables which is assigned the value of the offset address of the called code reading subroutine. After returning from the subroutine, the first argument will contain the BCII code read from the code buffer, if any, and the second argument will contain the status, i.e., the 0 value of the second argument indicates that no code was found during subroutine call and the 1 value of the second argument indicates that a code is passed from the code reading subroutine.

CHAPTER 7
RESULT, DISCUSSION AND CONCLUSION

7.1 RESULT AND DISCUSSION

The project was first initiated from the observation that the only available Bengali key-board in Bangladesh–the 'Optima Munir' key-board is completely unsuitable for computer applications. The need for developing a new Bengali key-board suitable for computer applications has been felt. Development of such key-board requires that the key-board should accomodate those characters by which Bengali script can be represented in linguistically acceptable form and the same characters can be represented in various soft-copy and hard-copy printing devices. The number of the characters should be such that the characters can be encoded by a standard size of code bits. Beside these, for the sake of standerdization, the key-board should be such that the same key-board can be used in mechanical and electric typewriter.

At the begining of the project life-cycle, the need for a linguistically acceptable theoretical framework for selecting the key-board primitives is felt. Such a theoretical framework is developed and the total character set used in Bengali script has been identified. The identified Bengali characters amount to 434 which is too much to be accomodated in a handy size of key-board as well as in various hard-copy printing devices. These 434 characters require 2 bytes for encoding them which is inconvenient from the view point of computer resource requirements. To make the developed system optimal, need for selecting a set of graphic symbols arises by which the 434 Bengali characters can be represented in the linguistically acceptable form. The selected set of graphic symbols should be such that the graphic symbols can be accomodated on a handy size of key-board and the same graphic symbols can be represented in various soft-copy and hard-copy printing devices. The number of the selected graphic symbols should also be such that the selected graphic symbols can be encoded by a standard size of code bits.

For selecting the desired graphic symbols set, the first approach has been made with the view that the Bengali characters are to be mapped under software control. It was found that a 172 graphic symbol set is required to map all the Bengali characters in their conventional letter press form. But a number of notable problems have been identified which makes this approach inconvenient. One of the notable problems of this approach is that this approach will not be convenient for real-time applications. Another notable problem will arise with this approach that representing these huge number of graphic symbols in various hard-copy printing devices and in mechanical and electric typewriters will be difficult.

To overcome the shortcomings of the software mapped approach, a direct method of realizing the Bengali characters is needed, i.e., a set of graphic symbols is required by which the 434 Bengali character set can be generated by concatenation of graphic symbols with a minimum of superposition, if that can not be removed completely. Selection of the graphic symbols should be such that the selected graphic symbols can be represented in various hard-copy printing devices. The symbol set should also consists of such number of graphic symbols that the symbols can be encoded with a maximum of 8-bit coding scheme. Working in this approach, it has been found that no standard is being maintained about the shape of the Bengali characters in various letter press printing systems. On the other hand, the shape of these characters in lino-type is completely different from that of letter press printing. For selecting the desired graphic symbol set, need for the standardization of shape of the Bengali characters is felt. As no such standard is found, for the sake of the present work, shape for each of the 434 Bengali characters has been assumed. About more than 90% of the assumed graphics are similar to that presently used in lino-type. For the sake of simplicity and ease of computer implementation, shapes of some of the less frequent unusual shaped compound byanjana varnas have been simplified in the linguistically acceptable form. Based on this assumed graphics, a 131 graphic symbol set has been selected where each of the graphic symbols has its own lexical identity for providing the opportunity of lexical analysis by computer.

Based on the statistical analysis of frequency of occurrence of the 131 selected Bengali graphic symbols, two key-board lay-outs have been deviced- one with 56 main keys and other with 47 main keys. The key-board lay-out with 56 main keys has been deviced with the aim to implement with a new hardware design and the key-board lay-out with 47 main keys has been deviced for adapting the existing english key-board of the widely used microcomputers as Bengali key-board. Placement of the graphic symbols on the key-board is so made that the load is equally distributed on all the active fingers and at the same time the key-board lay-out provides some logical ordering of the symbols such that remembering the position of the symbols can be guided by same logical manner.

All the 131 Bengali graphic symbols can easily be represented in VDUs and dot matrix printers. But the problem arises with the Line printers and Daisy-wheel printers. Fortunately, a set of 95 impression symbols has been selected by which all of the Bengali graphic symbols, except rarely used #,&, @ and _ special graphic symbols, can adequately be generated if superposition is allowed. These 95 impression symbols with a space totalling 96 symbols can be represented in typically used 96 character Line printers and Daisy-wheel printers. In Daisy-wheel printers, the speed penalty will not be much pessimistic because the average no. of superposition/character is statistically estimated to be 1.3205. But the speed penalty for a Line printer with 132 character line will be much high. It has been statistically estimated that on average 3.0103 passes will be required for each line.

The 131 Bengali graphic symbols along with Space, Delete and 32 ASCII standard control codes have been encoded using 8-bit coding scheme. In serial data communication systems with capability of handling 8-bit characters with additional parity setting, this 8-bit code can be used directly. In serial data communication system, which handle 7-bit character with parity setting at the 7th bit and in parallel data communication systems where the 7th bit is used for parity setting, a code mapping scheme has been proposed such that the parity can be set for error checking. It has been statistically estimated that, in this code mapping scheme, average bits per character will be 8.2815 which is infact less than 9 bits which would be required in those systems which handle 8-bit character with additional parity setting.

As no industrial back-up is still available in Bangladesh to produce new Bengali key-board suitable for computer applications, the Adapted BCII key-board lay-out has been deviced dor adapting the existing english key-board with 47 main keys available with all available microcomputers as Bengali key-board. The key configuration of these key-boards allow to implement the Adapted BCII key-board lay-out with these key-boards, but the nature of the key-board operation and the code generation mechanism are required to be completely changed from the existing key-board operation and the code generation mechanism. As the available key-board handling routine cannot be used for the present purpose, an algorithm is developed to make the key-board suitable for adaptation as Bengali key-board. This algorithm bypasses the original key-board handling routine of the concerned microcomputer and changes the nature of the key-board operation as needed by the ABCII key-board for generating the BCII codes from the adapted key-board. Implimentation of this algorithm provides the opportunity of adapting the existing key-board of any available microcomputer without changing any hardware configuration. This algorithm has been experimented with IBM PC microcomputer. The IBM PC is chosen for the present experimentation because a number of microcomputers, manufactured by other manufacturers, are available which are compatable with the IBM PC. The selection of the IBM PC will provide the opportunity of implementing the developed routine based on the developed algorithm in a variety of microcomputers. Two key-board handling routines have been developed-one in non-interrupting mode and the other in interrupting mode. The interrupting mode routine is varsetile, flexible and suitable for large applications and has been written in assembly language with providing the provision that the routine can be called from any processing routine, written in BASIC language, according to some set rules.

7.2 FUTURE SCOPE OF WORK

Based on the developed key-board lay-out and the developed key-board handling algorithm for adapting the existing microcomputer key-board as Bengali key-board, the following future research works can be carried on:

i) Bengali Information Processing systems can be developed using the developed key-board.

ii) A complete hardware, including key-board processor, serial to parallel data conversion and interrupt management support as required, can be developed with required software support for the BCII key-board lay-out having 56 main keys such that the developed system can be interfaced with available microcomputer systems.

iii) Software routines can be developed for printing the Bengali text in typically used 96 character Line printers and Daisy wheel printers using the 95 Bengali Impression Symbol Set.

iv) Software routines can also be developed for printing Bengali text in dot matrix printer and VDUs in graphics mode using the 131 Bengali Graphic Symbol Set.

v) Hardware as well as softare interface can be developed for the BCII code mapping scheme for the data communication systems which handle a 7-bit character.

7.3 CONCLUSION

The key-board lay-outs have been deviced with the expectation that each and every key-stroke will produce a uniquely identifiable graphic symbol having lexical identity as well as numeric code. All compound byanjana varnas and aksharas are to be generated by concatenating the constituent Bengali graphic symbols and superposing the ্য-fala only. More than 90% of the generated characters will be similar to that of the Lino-type presently practiced. The graphics of only a few number of less frequent characters have been altered in the linguistically acceptable form and this little sacrifice of conventional graphics of these Bengali characters is introduced to make the developed system 'optimal' from the view point of implementation in all the fields of implementation concerning computer applications, because the selected graphic symbols can be represented in various typically used soft-copy and hard-copy printing devices and the graphic symbols have been encoded in an 8-bit coding scheme. Moreover, the selection of these graphic symbols and the key-board lay-outs makes it possible to adapt the existing english key-board of the available microcomputers as Bengali key-board without any additional support. The proposed key-board lay-outs have been deviced based on the frequency of occurrence of the Bengali graphic symbols to distribute the load equally on all the active fingers for enhancing the typing speed which makes the key-board lay-outs 'optimal' from the view point of typing speed. Algorithm and software routine has been developed for adapting the existing english key-board of the available microcomputers as Bengali key-board, which provides the opportunity of having a Bengali key board with any available microcomputer without changing any hardware configuration. This requires no industrial back-up for producing required hardware interface and makes the developed system 'optimal' from the view point of industrial involvement.

REFERENCES

1. Das, P.K.,"On Information Content Of Bengali Language And Noise In Microwave Communication In Bangladesh", M.Sc. Engg. Thesis, Deptt. of Electrical Engineering, BUET, Dhaka,1976.

2. Ghoshal, T.K.,G. Das, K.K. Datta, S. Mitra and S. Bhattacharya, "Vidyasagar- A Bengali-Ahamia Text Processing Attachment", Journal of the Institution of Electronics and Telecommunication Engineers, Vol. 30, NO.6, PP 190-195, India,1984.

Mitra, S., S. Bhattacharya and T.K. Ghoshal, "Representing Variable Width Composite Consonant Text In Character Mode Raster-Scan VDU", Journal of the Institution of Electronics and Telecommunication Engineers, Vol. 30, No. 6, PP 229-230, India, 1984.

4. Das, G.,S. Bhattacharya and S.Mitra, "Representing Ahamia, Bengali and Monipuri Text In Line Printer And Daisy-Wheel Printer", Journal of the Institution of Electronics and Telecommunication Engineers, Vol. 30, No. 6, PP 251-256, India, 1984.

5. Rahman, S.M.,M. Ahmed, "Bengali Alphanumerics For Digital Display", Convention of the Institute of Engineers, Bangladesh, 6-9th Jan., 1984.

Ahmed, M., "Design Of Bengali Alphanumeric Segment And Dot Matrix Display", M.Sc. Engg. Thesis, Deptt. of Electrical and Electronic Engineering, BUET, Dhaka, April, 1986.

7. Karim, A.N.M.M., "Variable Width Character Generator And Display Using Dot-Matrix", M.Sc. Engg. Thesis, Deptt. of Computer Engineering, BUET, Dhaka, 1986.

8. Datta, P., "Standardisation of Bengali Code Of Signs: Vidyasagar, Its Predecessors And Successors", Proc. 7th Conf., All-India Type Founders Federation, PP 23-34, Jan., 1982.

9. Ekeblad, F.A., "The Statistical Method in Business, applications of probability and inference to business and other problems", pp 123-184, John Wiley & Sons, Inc., New York, 1962.

10. Mostafa, M.G., "Methods of Statistics", pp 149-184, Anwar Publication, Dhaka, 1972.

11. James, M.L., G.M. Smith and J.C. Wolford, "Applied Numerical Methods for Digital Computation with FORTRAN and CSMP", 2nd edn., pp 89-165, Harper & Row, Publishers, Inc., New York, 1977.

12. Intel Corporation, "Component Data Catalog", Santa Clara, California, 1982.

13. Microsoft, "BASIC", IBM,1983.

14. Microsoft, "BASIC Compiler", IBM,1982.

15. Microsoft Corporation, "Microsoft Link: Linker Utility", Bellevue, WA, 1983.

16. IBM, "Disk Operating System", IBM, 1984.

17. Microsoft Corporation, "Microsoft Macro Assembler: Utility for 8086 and 8088 Microprocessors", Bellevue, WA,1983.

18. Microsoft Corporation,"Microsoft DEBUG: Utility for 8086 and 8088 Microprocessors", Bellevue, WA, 1983.

19. Intel Corporation, "iAPX 88 Book", Santa Clara, California,1981.

20. Azad, H. (chief editor), "Bangla Bhasha (The Bengali Language)", Vol. 1, Bangla Academy, Dhaka, 1984.

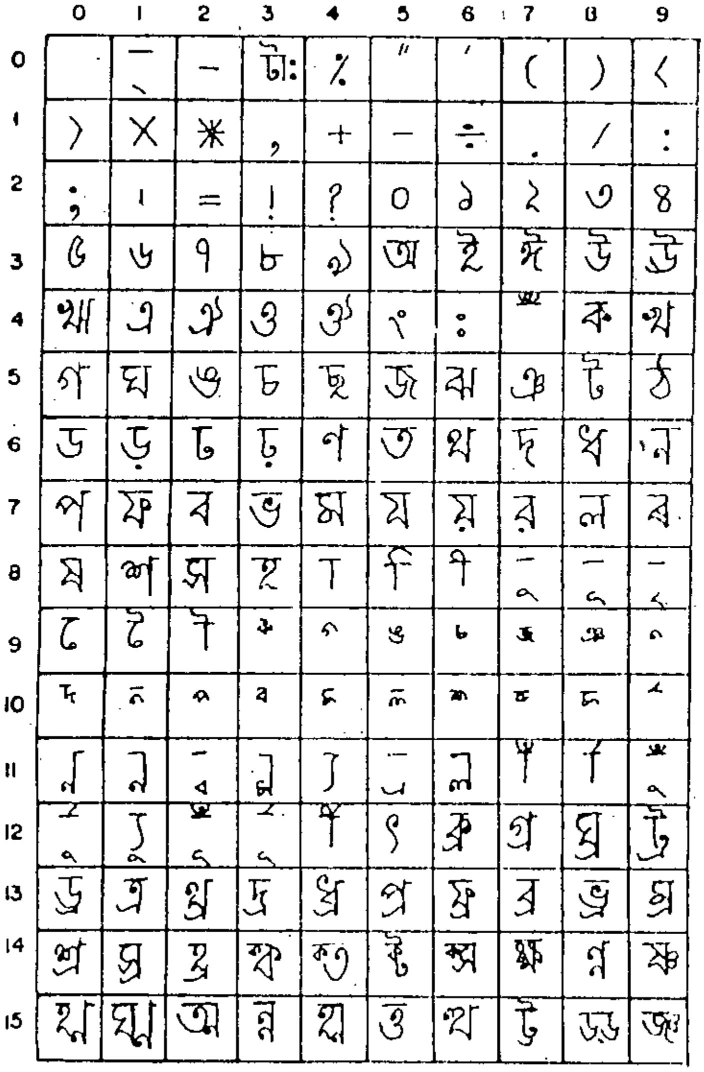
21. Shahidullah, M., "Bangala Byakarana (The Bengali Grammer)", Provincial Library, Dhaka, 1377 (Bang.).

22. Rashid, A.N.M.B., "Bangla Uchcharana Abhidhana (Dictionary of Bengali Pronunciation), Kousumi Sahitya Patra, Dhaka, 1979.

23. Choudhury, M., "Bangla Gadyareeti (Bengali Prose Style)", Bangla Academy, Dhaka,1970.

APPENDIX A
EXTENDED SEN AND DATTA GRAPHIC SYMBOL SET AND THEIR FREQUENCY OF OCCURRENCE

TABLE A-1: EXTENDED SEN AND DATTA GRAPHIC SYMBOL SET (SDBM)



	0	1	2	3	4	5	6	7	8	9
0		[illegible]	–	টা:	%	″	′	(	)	<
1	)	×	*	,	+	−	÷	.	/	:
2	;	।	=	!	?	০	১	২	৩	৪
3	৫	৬	৭	৮	৯	অ	ই	ঈ	উ	ঊ
4	ঋ	এ	ঐ	ও	ঔ	ং	ঃ	ঁ	ক	খ
5	গ	ঘ	ঙ	চ	ছ	জ	ঝ	ঞ	ট	ঠ
6	ড	ড়	ঢ	ঢ়	ণ	ত	থ	দ	ধ	ন
7	প	ফ	ব	ভ	ম	য	য়	র	ল	[illegible]
8	ষ	শ	স	হ	া	ি	ী	[illegible]	[illegible]	[illegible]
9	ে	ৈ	ৗ	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
11	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
12	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ক্র	গ্র	ঘ্র	ট্র
13	ড্র	ত্র	থ্র	দ্র	ধ্র	প্র	ফ্র	ব্র	ভ্র	ম্র
14	শ্র	স্র	হ্র	ক্ক	ক্ত	ক্ট	ক্ম	[illegible]	[illegible]	[illegible]
15	হ্ন	ঘ্ন	ত্ম	ন্ন	হ্ম	[illegible]	[illegible]	ট্ট	ড্ড	জ্ঞ

(72) = ব (141) = স্র etc.

TABLE A-2: FREQUENCY OF OCCURRENCE (ON THE BASIS OF 66,752 NO. OF OCCURRENCE) OF EXTENDED SEN AND DATTA GRAPHIC SYMBOL SET (SDBM) (Gourhari Das et al., 1984)

SDBM Seq.No.	No. of Occurrence*	% of Occurrence	SDBM Seq.No.	No. of Occurrence*	% of Occurrence
(01)	73	0.110	(31)	1	0.001
(02)	12	0.019	(32)	4	0.006
(03)	0	0.000	(33)	0	0.000
(04)	0	0.000	(34)	0	0.000
(05)	15	0.023	(35)	1,121	1.680
(06)	0	0.000	(36)	754	1.130
(07)	0	0.000	(37)	700	1.050
(08)	0	0.000	(38)	233	0.350
(09)	0	0.000	(39)	12	0.018
(10)	0	0.000	(40)	3	0.005
(11)	0	0.000	(41)	634	0.950
(12)	0	0.000	(42)	20	0.030
(13)	76	0.114	(43)	393	0.590
(14)	0	0.000	(44)	2	0.003
(15)	128	0.192	(45)	253	0.380
(16)	0	0.000	(46)	26	0.040
(17)	0	0.000	(47)	33	0.050
(18)	0	0.000	(48)	2,476	3.710
(19)	0	0.000	(49)	380	0.570
(20)	4	0.006	(50)	660	0.990
(21)	92	0.139	(51)	120	0.180
(22)	170	0.255	(52)	13	0.020
(23)	1	0.001	(53)	587	0.880
(24)	9	0.014	(54)	567	0.850
(25)	4	0.006	(55)	654	0.980
(26)	14	0.021	(56)	53	0.080
(27)	1	0.001	(57)	6	0.010
(28)	1	0.001	(58)	607	0.910
(29)	2	0.003	(59)	146	0.220
(30)	2	0.003	(60)	93	0.140

Table A-2 (continued)

SDBM Seq. No.	No. of Occurrence*	% of Occurrence	SDBM Seq. No.	No. of Occurrence*	% of Occurrence
			(95)	126	0.190
(61)	239	0.440	(96)	53	0.080
(62)	20	0.030	(97)	6	0.010
(63)	0	0.000	(98)	33	0.050
(64)	226	0.340	(99)	40	0.060
(65)	2,209	3.310	(100)	80	0.120
(66)	380	0.570	(101)	427	0.640
(67)	1,114	1.670	(102)	26	0.040
(68)	493	0.740	(103)	13	0.020
(69)	2,589	3.880	(104)	66	0.100
(70)	967	1.450	(105)	13	0.020
(71)	113	0.170	(106)	33	0.050
(72)	2,236	3.350	(107)	106	0.160
(73)	420	0.630	(108)	206	0.310
(74)	1,455	2.180	(109)	253	0.380
(75)	400	0.600	(110)	0	0.000
(76)	1,214	1.820	(111)	26	0.040
(77)	3,698	5.540	(112)	246	0.370
(78)	1,655	2.480	(113)	80	0.120
(79)	0	0.000	(114)	587	0.880
(80)	206	0.310	(115)	6	0.009
(81)	694	1.040	(116)	40	0.060
(82)	1,535	2.300	(117)	140	0.210
(83)	807	1.210	(118)	60	0.090
(84)	6,922	10.370	(119)	13	0.020
(85)	3,277	4.910	(120)	4	0.006
(86)	687	1.030	(121)	6	0.010
(87)	1,114	1.670	(122)	0	0.000
(88)	173	0.260	(123)	2	0.003
(89)	146	0.220	(124)	6	0.009
(90)	5,780	8.660	(125)	73	0.110
(91)	46	0.070	(126)	40	0.060
(92)	46	0.060	(127)	66	0.100
(93)	0	0.000	(128)	0	0.000
(94)	3	0.005			

TABLE A-2 (Continued)

SDBM Seq. No.	No. of Occurrence*	% of Occurrence	SDBM Seq. No.	No. of Occurrence*	% of Occurrence
(129)	26	0.040			
(130)	1	0.002			
(131)	126	0.190	(146)	6	0.010
(132)	0	0.000	(147)	120	0.180
(133)	46	0.070	(148)	2	0.003
(134)	0	0.000	(149)	2	0.003
(135)	293	0.440	(150)	2	0.003
(136)	4	0.007	(151)	0	0.000
(137)	6	0.010	(152)	6	0.009
(138)	6	0.009	(153)	53	0.080
(139)	5	0.008	(154)	0	0.000
(140)	60	0.090	(155)	80	0.120
(141)	4	0.007	(156)	0	0.000
(142)	0	0.000	(157)	4	0.007
(143)	6	0.010	(158)	4	0.006
(144)	100	0.150	(159)	26	0.040
(145)	0	0.000	SPACE	11,307	16.940

* Note: No. of occurrence has been computed from % of occurrence and total no. of occurrence.

APPENDIX B

FREQUENCY OF OCCURRENCE OF BENGALI CHARACTERS (Prabir Kumar Das, 1976).

TABLE B-1: FREQUENCY OF OCCURRENCE (ON THE BASIS OF 43,126 No. OF OCCURRENCE) OF BENGALI CHARACTERS (Prabir Kumar Das, 1976).

Character*	No. of Occurrence					% of occurrence
	Texts(29,725 occurrence)	Fictions (3,529 occurrence)	Technical Texts (8,872 occurrence)	Newspapers (1,000 occurrence)	Total (43,126 occurrence)	
অ	312	17	92	5	426	0.9878
আ	378	79	49	7	513	1.1895
ই	488	75	105	23	691	1.6023
ঈ	36	4	0	1	41	0.0951
উ	209	5	35	14	263	0.6098
ঊ	0	0	0	0	0	0.0000
ঋ	2	0	0	0	2	0.0046
এ	369	64	115	7	555	1.2869
ঐ	9	0	11	0	20	0.0464
ও	232	37	66	5	340	0.7884
ঔ	0	0	0	0	0	0.0000
া	3,538	503	846	144	5,031	11.6658
ি	1,689	282	634	36	2,641	6.1239
ী	340	29	81	21	471	1.0921
ু	94	6	91	194	385	0.8927
ূ	73	4	30	3	110	0.2551
ৃ	103	3	41	2	149	0.3455
ে	2,421	387	817	57	3,682	8.5378
ৈ	24	2	11	2	39	0.0904

Table B-1 (contd.)

Character*	No. of Occurrence					% of occrrence
	Texts(29,725 occurrence)	Fictions (3,529 occurrence)	Technical Texts (8,872(occurrence)	Newspapers (1,000 occrrence)	Total (43,126 occrrence)	
ো	479	21	145	21	666	1.5443
ৌ	46	2	9	1	58	0.1345
ক	1,524	263	389	30	2,206	5.1152
খ	218	16	107	1	342	0.7930
গ	274	23	63	7	367	0.8510
ঘ	33	13	24	0	70	0.1623
ঙ	30	0	0	0	30	0.0696
চ	200	24	24	6	254	0.5890
ছ	260	20	49	7	336	0.7791
জ	343	31	105	15	494	1.1455
ঝ	21	6	6	0	33	0.0765
ঞ	0	0	0	0	0	0.0000
ট	227	37	135	19	418	0.9693
ঠ	65	10	22	0	97	0.2249
ড	30	11	0	8	49	0.1136
ঢ	20	6	3	0	29	0.0672
ণ	218	5	102	1	326	0.7559
ত	1,380	88	360	8	1,836	4.2573
থ	262	22	45	3	332	0.7698
দ	607	53	185	15	860	1.9942
ধ	171	14	58	3	246	0.5704
ন	1,392	131	300	38	1,861	4.3153

Table B-1 (Contd.)

Character *	No. of Occurrence					% of Occurrence
	Texts (29,725 occurrence)	Fictions (3,529 occurrence)	Technical Texts (8,872 occurrence)	Newspapers (1,000 occurrence)	Total (43,126 occurrence)	
প	698	37	282	27	1,044	2.4208
ফ	43	9	15	1	68	0.1577
ব	1,088	178	271	29	1,566	3.6312
ভ	187	16	43	15	261	0.6052
ম	924	107	119	16	1,166	2.7037
য	309	25	97	2	433	1.0040
র	2,123	258	518	47	2,946	6.8312
ল	765	69	175	12	1,021	2.3675
শ	287	32	134	9	462	1.0713
ষ	99	2	60	3	164	0.3803
স	669	42	151	19	881	2.0429
হ	639	26	146	10	821	1.9037
য়	663	58	198	20	939	2.1773
ড়	123	20	71	0	214	0.4962
ঢ়	3	0	0	0	3	0.0070
ৎ	40	1	68	1	110	0.2551
ং	0	0	0	0	0	0.0000
ঃ	35	2	11	0	48	0.1113
ঁ	59	11	4	2	76	0.1762
্	47	3	12	2	64	0.1484
ক্ত	47	2	55	0	104	0.2412
ক্ষ	93	5	34	5	137	0.3177
ঙ্খ	2	0	0	0	2	0.0046

Iable B-1 (contd.)

harac-er*	No. of Occurrence					% of occurrence
	Texts (29,725 occurrence)	Fictions (3,529 occurrence)	Technical Texts (8,872 occurrence)	Newspapers (1,000 occurrence)	Total (43,126 occurrence)	
ঙ্ঘ	0	0	1	0	1	0.0023
চ্চ	11	0	0	0	11	0.0255
চ্ছ	18	0	0	0	18	0.0417
জ্জ	4	0	0	0	4	0.0093
জ্ঞ	15	2	12	0	29	0.0672
ঞ্চ	15	0	1	0	16	0.0371
ঞ্জ	13	6	2	0	21	0.0487
ট্ট	0	0	0	6	6	0.0139
ণ্ড	37	0	0	0	37	0.0858
ত্ক	7	0	7	0	14	0.0325
ত্ত	4	6	0	0	10	0.0232
ত্থ	27	0	13	0	40	0.0928
ত্ন	2	0	1	0	3	0.0070
ত্ম	23	1	6	0	30	0.0696
ত্ব	12	8	0	0	20	0.0464
থ্ব	12	0	2	0	14	0.0325
দ্গ	7	0	0	0	7	0.0162
দ্ঘ	4	0	5	0	9	0.0209
দ্দ	29	1	4	0	34	0.0788
দ্ধ	4	0	6	0	10	0.0232
দ্ব	11	0	14	0	25	0.0580
দ্ভ	2	1	0	0	3	0.0070
দ্ম	23	0	2	0	25	0.0580

Table B-1 (contd.)

Charac-ter*	No. of Occurrence					% of occurr-ence
	Texts(29,725 occurrence)	Fictions (3,529 occurr-ence)	Technical Texts (8,872 oc-currence)	Newspapers (1,000 occurrence)	Total (43,126 occurr-ence)	
ল্ল	11	1	4	0	16	0.0371
শ্চ	13	0	3	0	16	0.0371
ষ্ক	5	0	5	0	10	0.0232
ষ্ট	46	5	18	0	69	0.1600
ষ্ঠ	33	0	1	0	34	0.0788
ষ্প	8	0	0	0	8	0.0186
ষ্ফ	13	0	0	0	13	0.0301
স্ট	0	0	1	0	1	0.0023
স্ত	29	1	19	0	49	0.1136
স্থ	32	6	13	0	51	0.1183
স্ন	2	0	3	0	5	0.0116
স্প	4	3	4	0	11	0.0255
স্ফ	4	0	1	0	5	0.0116
স্ব	5	4	0	0	9	0.0209
স্ম	3	0	2	0	5	0.0116
হ্ণ	3	0	0	0	3	0.0070
হ্ন	5	2	2	0	9	0.0209
হ্ব	9	0	6	0	15	0.0348
হ্ম	8	0	1	0	9	0.0209
হ্য	22	2	7	0	31	0.0719
হ্ল	2	0	0	0	2	0.0046
ক্ষ্ম	7		16	0	27	0.0626

Table B-1 (contd.)

Character*	No. of Occurrence					% of occurrence
	Texts (29,725 occurrence)	Fictions (3,529 occurrence)	Technical Texts (8,872 occurence)	Newspapers (1,000 occurrence)	Total (43,126 occurrence)	
ত্রু	16	3	6	0	25	0.0580
ত্র	27	2	0	0	29	0.0672
ঙ্ক	4	0	2	0	6	0.0139
ম্ল	3	0	1	0	4	0.0093
ন্ত্র	23	2	12	0	37	0.0858
ক্ষ্ম	2	0	0	0	2	0.0046
স্তু	4	0	17	0	21	0.0487
⅃	136	18	119	10	283	0.6562
ʃ	339	19	163	10	531	1.2313
ᓗ	243	15	139	8	405	0.9391
ঙ্গ	23	3	18	0	44	0.1020
রু	15	3	0	0	18	0.0417
রূ	12	0	26	0	38	0.0881
শু	31	2	0	0	33	0.0765
হু	32	8	4	0	44	0.1020
সৃ	0	0	5	0	5	0.0116
ক্র	4	0	10	0	14	0.0325
ন্তু	37	5	6	0	48	0.1113
সু	10	0	21	0	31	0.0719
—	416	28	128	16	588	1.3634
।	298	108	102	5	513	1.1895
;	154	8	49	2	213	0.4939
-	112	6	62	7	187	0.4336
:	63	0	11	0	74	0.1716
‘ ’	60	0	29	0	89	0.2064

Table B-1 (contd.)

Character*	No. of Occurrence					% of occurrence
	Texts (29,725 occurrence)	Fictions (3,529 occurrence)	Technical Texts (8,872 occurrence)	Newspapers (1,000 occurrence)	Total (43,126 occurrence)	
" "	25	0	3	0	28	0.0649
?	18	0	4	0	22	0.0510
,	12	48	127	12	199	0.4614
!	2	2	7	0	11	0.0255

*Note: Space has not been counted as a character.

APPENDIX - C

OPTIMA MUNIR KEY-BOARD LAY-OUT

FIG. C -1 OPTIMA MUNIR KEY-BOARD LAY-OUT

APPENDIX-D

THE ASCII CHARACTER CODE

TABLE D-1 American Standard Code for Information Interchange (ASCII), Standard No. X3.4-1968 of the American National Standards Institute.

		$b_6b_5b_4$ (column)							
$b_3b_2b_1b_0$	row (hex)	000 0	001 1	010 2	011 3	100 4	101 5	110 6	111 7
0000	0	NUL	DLE	SP	0	@	P	`	p
0001	1	SOH	DC1	!	1	A	Q	a	q
0010	2	STX	DC2	"	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB	'	7	G	W	g	w
1000	8	BS	CAN	(	8	H	X	h	x
1001	9	HT	EM	)	9	I	Y	i	y
1010	A	LF	SUB	*	:	J	Z	j	z
1011	B	VT	ESC	+	;	K	[	k	{
1100	C	FF	FS	,	<	L	\	l	|
1101	D	CR	GS	-	=	M	]	m	}
1110	E	SO	RS	.	>	N	^	n	~
1111	F	SI	US	/	?	O	_	o	DEL

Control Codes

NUL	Null	DLE	Data link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronize
BEL	Bell	ETB	End transmitted block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete or rubout