AN UNIVERSAL CODING SYSTEM FOR MULTI-LINGUAL ENVIRONMENT

S.S. TSENG, C.C. Hsieh,
JACK K.T. HUANG, C.T. CHANG
and C.C. YANG

CHINESE CHARACTER ANALYSIS GROUP
COUNCIL FOR CULTURAL PLANNING
AND DEVELOPMENT
TAIPEI, REPUBLIC OF CHINA

Abstract

In this paper. An interchange coding system of 3-byte/ 2-byte/ 1-byte graphic symbols for multi-lingual software environment will be presented. The structure of this system is based on CCCII (Chinese Character Code for Information Interchange) developed in 1980. The first version of this system was completed in 1983.

The version 3 of CCCII is flexible enough to adopt any existing 3-byte/ 2-byte/ 1-byte coding systems that are based on ISO-646 and -2022. At this moment, it has space to accommode more than 90 7-bit bytes standards. Therefore, most of the published 7-bit byte standards may be integrated into this system without system modification. For Oriental languages, the character sets collected include those used in R.O.C., Japan, Korea and mainland China. There are 42,000 regular Chinese characters in this version 3 character set. Besides, the variations for the regular characters collected are beyond 11 thousands. JIS C 6226 and GB 2312 character set are also located in the version 3 of CCCII. A mechanism of interchange among these character sets is provided within the system.

The 3-byte structure of CCCII provides a convenient mechanism for cross mappings among different 3-byte/ 2-byte/ 1-byte character sets. Used in conjunction with the attribute data base of characters, the CCDB (Chinese Character Data Base), this system provides attribute cross-reference access features among different 3-byte/ 2-byte/ 1-byte standards. For example, an user can have access to Japanese Kanji through the index mechanism of CCDB of corresponding Chinese character.

At present, the CCCII system structure is fixed but its symbol set is still open. We expect that in the later part of 1986 or in early 1987 the characters collected in the version 4 of CCCII will exceed 70 thousands. This CCCII coding system is not only beneficial for international information interchange, but also valuable for developing multi-lingual software in tommorrow's computer system.

Introduction to Multi-lingual Environment

The multi-lingual environment is based on CCCII coding system [1] and CCDB [2] which issued in 1980 and 1982 respectively. CCCII is a three 7-bit-byte code based on ISO-646 and -2022, the graphic area of this system has 94*93*94, totally 821,748 coding positions and 94 control codes for data processing. The first byte, B1, denotes the positions within a section, the second byte, B2, denotes the sections of a plane, and the third byte, B3, denotes the planes, as shown in figure 1. CCDB is a data base for Chinese characters and their associated attributes with a set of necessary software tools. The software structure and index mechanism of CCDB are shown in figure 2-3 respectively. Under those structures of CCCII and CCDB, it is not only adequately used for more than 70 thousands Chinese characters, but also adequately used for the Oriental languages which use a lot of Chinese characters or Chinese-likely characters, such as in Japan and Korea [3].

In fact, we need a multi-lingual coding system for international imformation processing systems [4], here with the CCCII and CCDB provide an excellent environment for the work. The advantages are explained as follows:

(1) The implementation of CCCII is shown in fig 4, 94 planes devide into 16 layers, each layer has 6 planes except layer 16 has only 4 planes. The first section of every plane is reserved for CCCII control code which will explain later, section 2 to 15 of the first plane of each layer are reserved for user area except layer 13, 14, 15, and 16. Please see Fig. 5 and 6.

(2) The first layer allocates the regular form of Chinese characters. The second layer through layer 12 allocate the variant forms of Chinese characters and the simplified forms are treated as variant forms which are allocated in layer 2 only.

(3) We treat the 6,798 Kanji of JIS C 6226 as variant forms of chinese characters and allocate them in layer 13. The other symbols of JIS C 6226 then allocated in the section 2 to 15 of layer 13. By the mechanism of CCDB, the Japanese Characrters can be coded in CCCII structure in such a way, and we had done it already.

(4) Layer 14 is reserved for Korean characters in the same way as allocate the Japanese characters in layer 13.

(5) Plane 85, the first plane of layer 15 is reserved for characters of minority Chinese langueages such as Mongolian, Tibetan, Manchurian, Moslem, etc.

(6) Plane 86 through plane 94 are reserved for the other languages of the world. There are adequately enough coding spaces and convenience mechanisms for an universal lauguage coding system.

(7) The control codes of CCCII include two parts, the first part of them are used for the extension techniques by means of escape sequence[5]. The second part of them are used for edition or other applications of text files. These codes keep B3 = blank, B2 = 33 as control code indicator, practically, only B1 is used. We have assigned 27 control codes as shown in fig. 7, it can be inreased by pratice requirement. Those codes are grouped as follows:

(a) Text Separators:
DPA (Delimiter between Parts)
DCH (Delimiter between Chapter)
DSE (Delimiter between Section)
DBL (Delimiter between Blocks / Subsections)
DPR (Delimiter between Paragraphs)
DST (Delimiter between Sentences)
DWO (Delimiter between Words)
DTI (Delimiter between Titles)
DCU (Delimiter between Chuns (rolls))
DVO (Delimiter between Volumns)
DPG (Delimiter between Pages)
DLN (Delimiter between Lines)

(b) Typesetting Effectors:
SCH (SKip one Character)
SLN (SKip one Line)
CHS (Character Scaling)
CSR (Character Scale Recovery)
CFR (Character-Font Recovery)
CDR (Character-Display Reverse)
CHL (Character High-light)
HLR (High-light Recovery)

(c) Code-Format Switchers:
PSL (Plane-Shift, Locked)
PSR (Plane-Shift, Recovery)
PSU (Plane-Shift, Unlocked)
SSL (Section/plane-shift, Locked)
SSU (Section/plane-shift, Unlocked)
HOM (Home)

Now, we will introduce a mechanism of universal coding system as below.

The mechanism for universal coding system

1. Intoduction to the CCDB

Chinese Character Data Base (CCDB), shown as fig. 2, is a data base together with necessary software that was developed based on the structure of CCCII in order to make the application of CCCII easy and efficient. CCDB has three main fuctions:

(1) To provide Chinese charater constructions, phonetic transcription, dot matrix configurations of Chinese characters, various correlated input methods and various existed corresponding Chinese character coding systems [5], please see figure 8.

(2) To provide controlling and searching mechanism of transcription table between corresponding lexicographic character fonts and phonetic symbols.

(3) To provide the interchanging function among various existed corresponding multi-bytes coding systems, for example, from 3-byte CCCII to 2-byte JIS C 6226 or from 2-byte JIS C 6226 to 3-byte CCCII, etc.

The CCDB file structure is mainly organized by two portions: data files and indexing files, The data files subdivided into 16 layers. All files are grouped into seven file-groups as described below.

(1) G1 (files of layer 1): designates the data files of regular form of Chinese characters.

(2) G2 (files of layer 2): designates the data and searching files of simplified forms of variant forms of Chinese characters.

(3) G3 (files of layer 3-12): designates the data and searching files of other variant forms of Chinese characters appearing in layer 1. Currently, only five selected variant forms are vertically allocated from layer 3 to 7.

(4) G4 (files of layer 13): the data and searching files of Japanese Kanas and 6,798 Kanji of JIS C 6226 are allocated in layer 13.

(5) G5: Inverted lists of Chinese lexiographic fonts, phonetic symbols, and the other attributes of Chinese Characters.

(6) G6: Indexed tables of various Chinese character input methods, e.g., 3-corn code, Dragon input method, etc.

(7) G7: Cross-reference tables of various existed relevant coding systems.

The Chinese Characters stored in CCDB are hired R94, a 16-bit condensed code of CCCII.. The file groups separated by a layer pointer LP. To calculate R94 and LP, CCDB provide formalas as

R94=[(B3-33) mod 6]*94*94+(B2-34)*94+(B1-33), and
LP= (B3-33)/6.

The searching mechanism of CCDB uses three different addressing modes: (see Figure 3)

(1) Direct addressing: If LP=0, the R94 code is the storing address of the regular form of Chinese Characters in layer 1 in the environment of CCCII.

(2) Indirect addressing: If LP=1, the R94 code is the address of the storing address of the simplified form of Chinese characters in layer 2. If LP=12, R94 code is the indirect address of Japanese Kanas and Kanji of JIS C 6226 allocated in layer 13 in the environment of CCCII. If LP=13, the R94 code is the indirect address of KIPS or KSC-5619 Korean characters which to be allocated in the environment of CCCII.

(3) Indirect-displacement addressing: If LP=2 to 11, the R94 code is the address of a Base, denoted by B, and the storing address of the variant forms of Chinese charactes in layer 3 through layer 12 in the environment of CCCII are calculate by the formala m = B+LP-2, where LP must less than or equal to the upper boundary of LP (L). The upper boundary of LP and B are different from character to character, so it should be pre-determinated when the character was stored.

The above mentioned three addressing modes are also applicable to currently reserved 85-94 planes of CCCII. The CCDB provides this mechanism for multi-lingual data procesing as an universal coding system as long as those languages were coded whithin the scope of CCCII. One important and common feature of CCCII and CCDB we would like to point out that the both mentioned systems are open systems in nature, subject to increment of appending of relevant searching and data files whithout further modification of the systems.

2. Code-format Switcheers:

Since CCDB is based on 3-byte CCCII. For up-grade the efficiency of data communication and storage, the 3-byte code can be transfer to 2-byte or 1-byte code in accordance with the usage of control codes defined by CCCII[6]. Those set of control codes are called code-format switchers. CCDB provides the control processing algorithm of code-format switchers as shown in Appendix A.

3. Introduction lto the CCCII-machine:

For the goal of implementation of universal coding system, the

CCCII-machine was designed that based on the structure of CCCII. The CCCII-machine system constructed by three main components, namely; (1) Kernel, (2) Transmission Unit and (3) Controller. Please see figure 9.

(1) Kernel: CCDB is the kernel in fact. By means of the proper routine calls, the imformation of CCDB will be accessed. Some flags tell the states of activities.

(2) Transmission unit: The transmission unit consists of the CORE (COde REceiver) and COTRA (COde TRAsmitter). It serves as an interface between CCDB and data communication devices. The CORE and COTRA are constructed by specific data structure and routines. Under the proper routine calls, the CORE and COTRA will be activated. Please refer to figure 10 and 13.

(3) Controller: Controller is the principal control program of CCCII-Machine. Because CCCII-Machine may implement on terminal, PC, or host computer, the controller then provide different functions in the different places of different applications. So the controller is application dependence, we will not go far about it then.

4. CORE (COde REceiver):

Refer to figure 10, the CORE recieves a character stream sending from the data transmission device. The character stream usually is an mixture including the control codes of ISO-646, the control codes of CCCII, the 3-byte and/or 2-byte and/or single byte codes of CCCII. The responsibilities of CORE are to pick up those who and give them the proper treatment and process, especially, the single byte or 2-byte codes should be transfer back to 3-byte representations. The data structure of CORE is explained as follows:

RI: 7-bit input register.
RB3: 7-bit register stores B3 of CCCII.
RB2: 7-bit register stores B2 of CCCII.
RB1: 7-bit register stores B1 of CCCII.
SD3: A stack to keep the B3 default values for PSL/PSU/PSR control codes under 2-byte condition.
SP: Stack point for SD3.
RD3 and RD2: 7-bit registers store B3 and B2 dafault values for SSU code under 1-byte condition.
S: State register, value 0 indicates the 3-byte condition, value 1 indicates the 2-byte condition and value 2 indicates the 1-byte condition.
C: Modula 3 register for byte position control, if 0 indicates next byte into RI is B3.

1 indicates next byte into RI is B2.
2 indicates next byte into RI is B1.
F: Register for error status.

The routine componets of CORE composed of a main control routine and a group of correlated subroutines. The flowchart of main control routine is shown in figure 12. Main control routine is an infinite-loop by nature. Once controller starts, the main control routine of CORE will ceaselessly operate.

The main control routine is also served as an interrupt-driven routine. It is liable to idle after accepted a new byte and processed. It will wake-up until next byte enters the RI. To stop the main control routine of CORE must be operated by means of system break. The details of design of idle, wake-up and break subject to implement-dependent manufacturers' and users' wish.

Error-handling functions are included in each subroutine. In case of error occurs, the status of CORE will be handled by F register of CORE, the the controller will take care after. A set of routines which wrote in pascal is shown in Appendix B.

5. COTRA (COde TRAnsmitter):

Same as CORE, the COTRA is constructed by two components; the data structure and the routines as shown in figure 13. In the data structure, except the S is defined slight differently from CORE, the definitions of RB3, RB2 and RB1 are the same as in CORE.

S is designated to control the number of bytes of CCCII to transmit out. When Controller calls COTRA to be requested to transmit CCCII. Functions of S are as follows:

(1) If S=0, then transmit all the RB3, RB2 and RB1.
(2) If S=1, then transmit the RB2 and RB1.
(3) If S=2, then transmit the RB1.

It is necessary to utilize routine calls to modify the contents of S. If the content of S is changed, COTRA will transmit corresponding control codes and parameter (if needed), as shown in figure 14.

There is no main control routine for COTRA routines, because COTRA is totally passive and completely controlled by Controller. A set of programs for COTRA routines which wrote in PASCAL is shown in Apendix C.

Conclusion

The CCCII-machine is not merely a hypothetic system on a drawing board, with its complete structure design and necessary detailed data it is ready for manufacturing. The CCCII-machine can be implemented on terminals, PC, or host computers. There are several manufacturers in Taiwan already taking place in developing the CCCII-machine. It is expected to see them on the market at end of this year, if every thing goes well.

Since CCCII provides the largest coding spaces (821,748 characters) available for data processing purpose. The 3-byte structure of CCCII allows for the multi-lingual data processing. The internal logic of 3-byte CCCII makes possible the linkage of regular, simplified, variant forms of Chinese character, and others such as Japanese Kana and Kanji, Korean Hangual, and almost all alphabetic characters to be processed all under one unique coding system. To avoid of multiple character sets in processed all under one unique coding system. To avoid of multiple character sets in processing of mulit-lingual environment, invoke escape sequences between them, and thus arbitrarily partitioning a unitary coding system.

The mechanism of CCCII-machine offers a single, coherent coding system and extends the applications limit for multilingual data processing in all fields, naturally, the application of library automation is included.

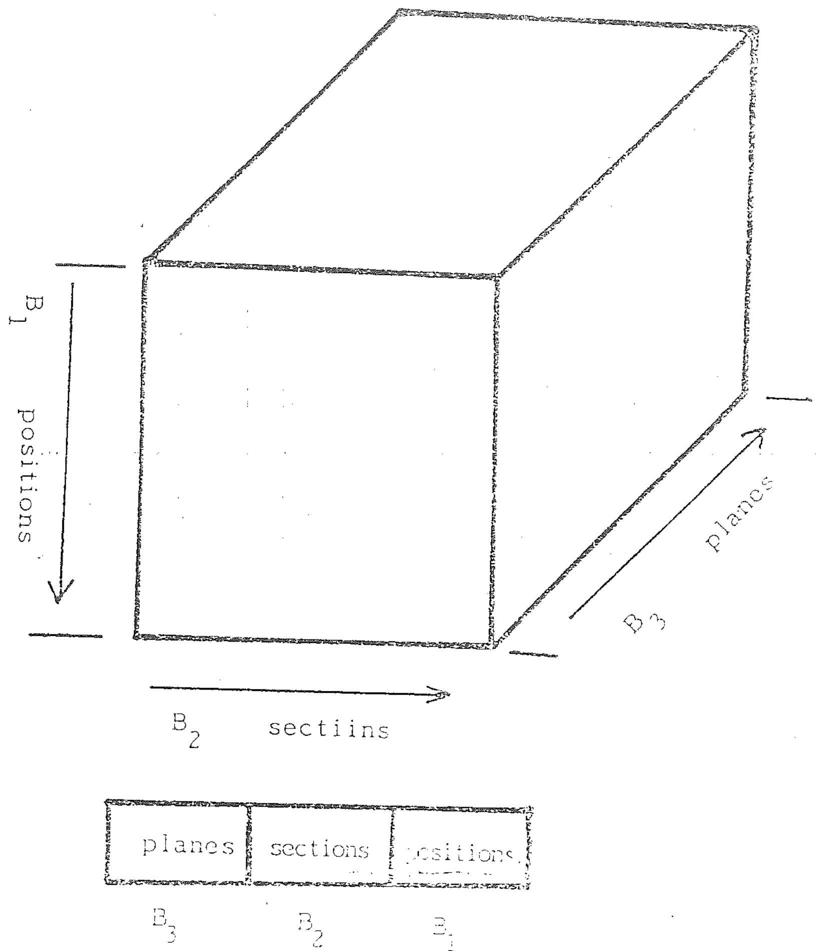


Figure 1. Coding structure of CCCII

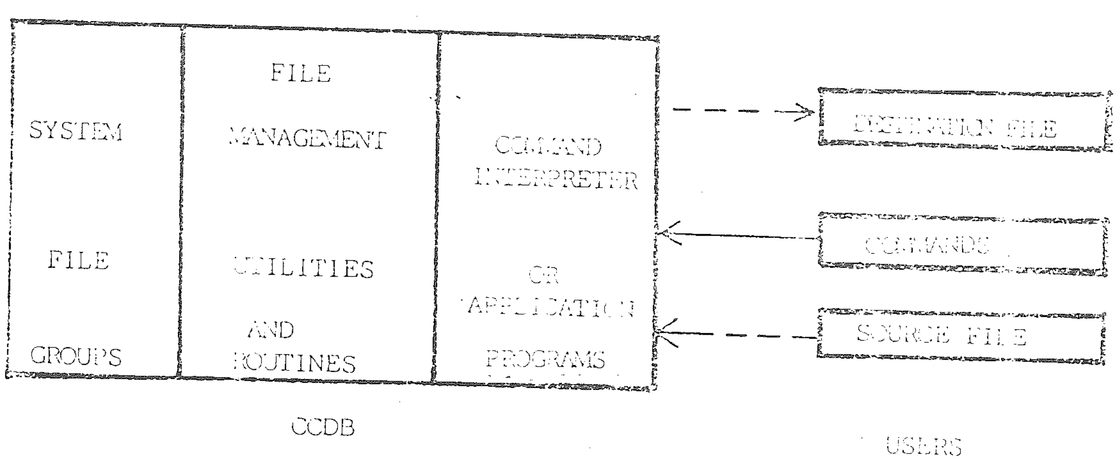


Figure 2. Software Structure of CCDB

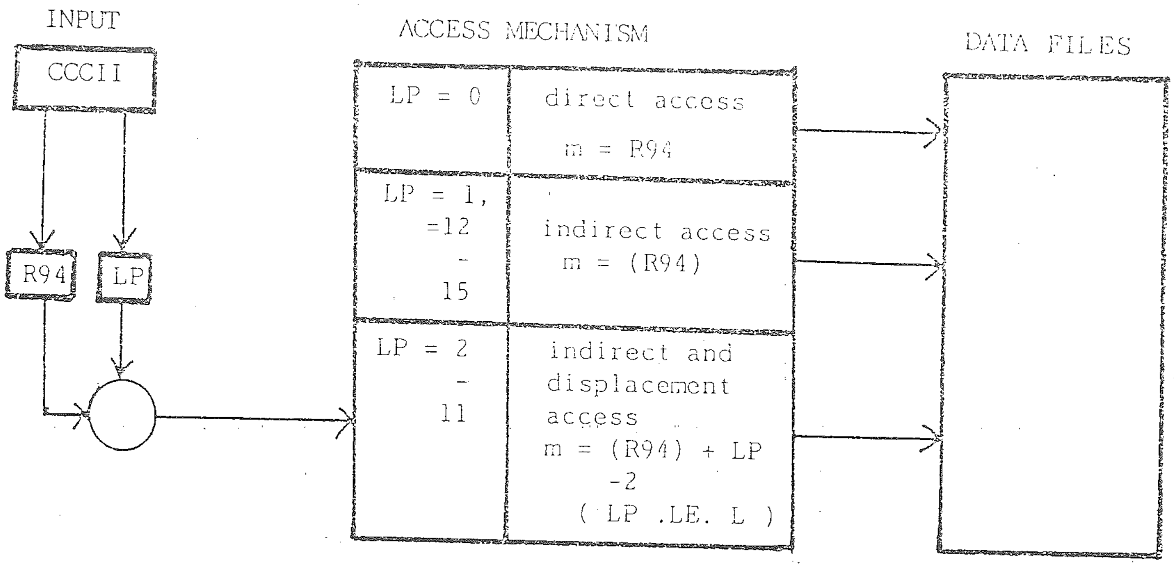
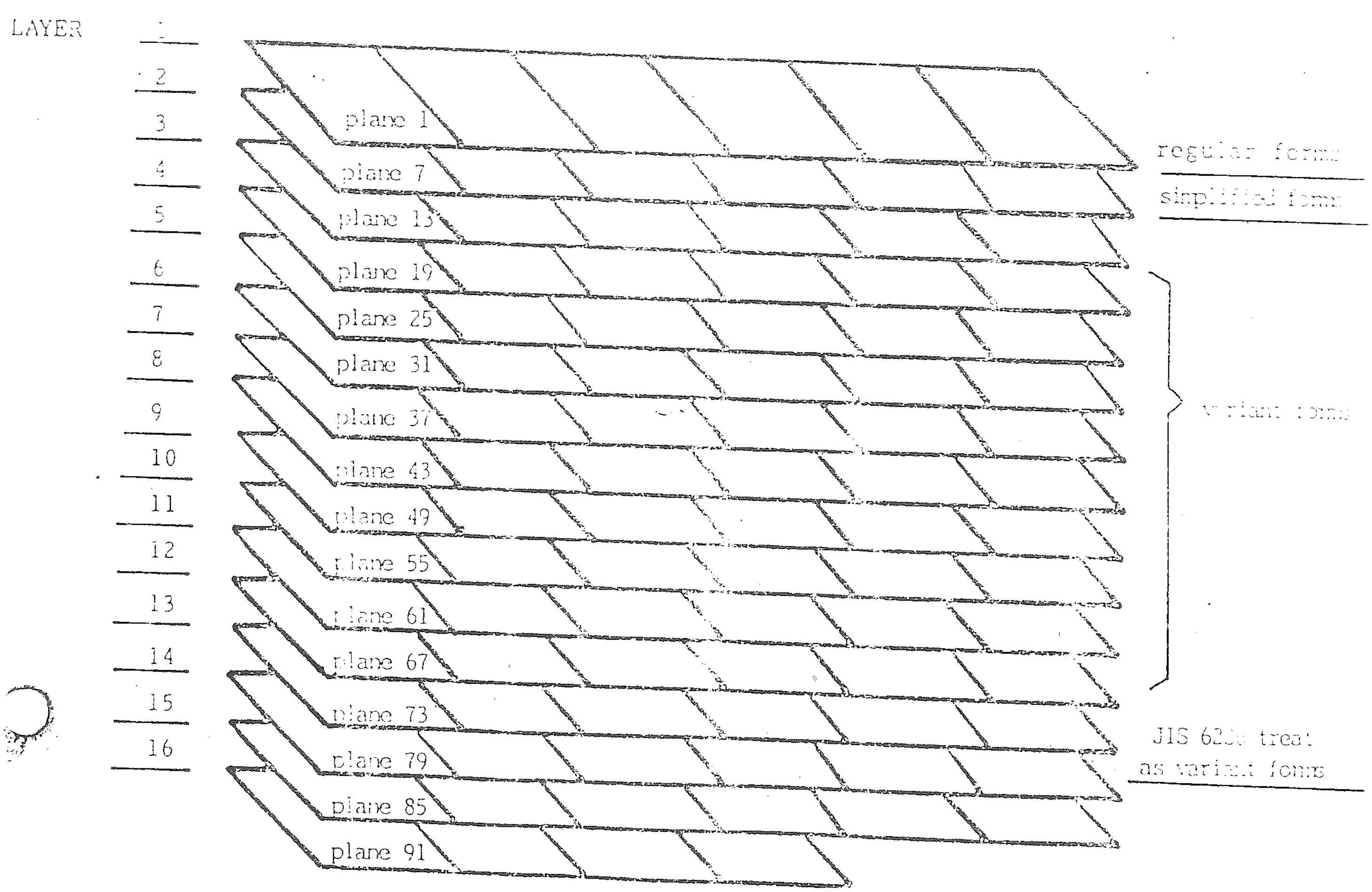


Figure 3. THE ACCESS MECHANISM OF CCDB
LP: Layer indicator, from 0 to 15
L: The upper boundary of LP
m: The effective address of the character to be access
R94: Condensed code of CCCII



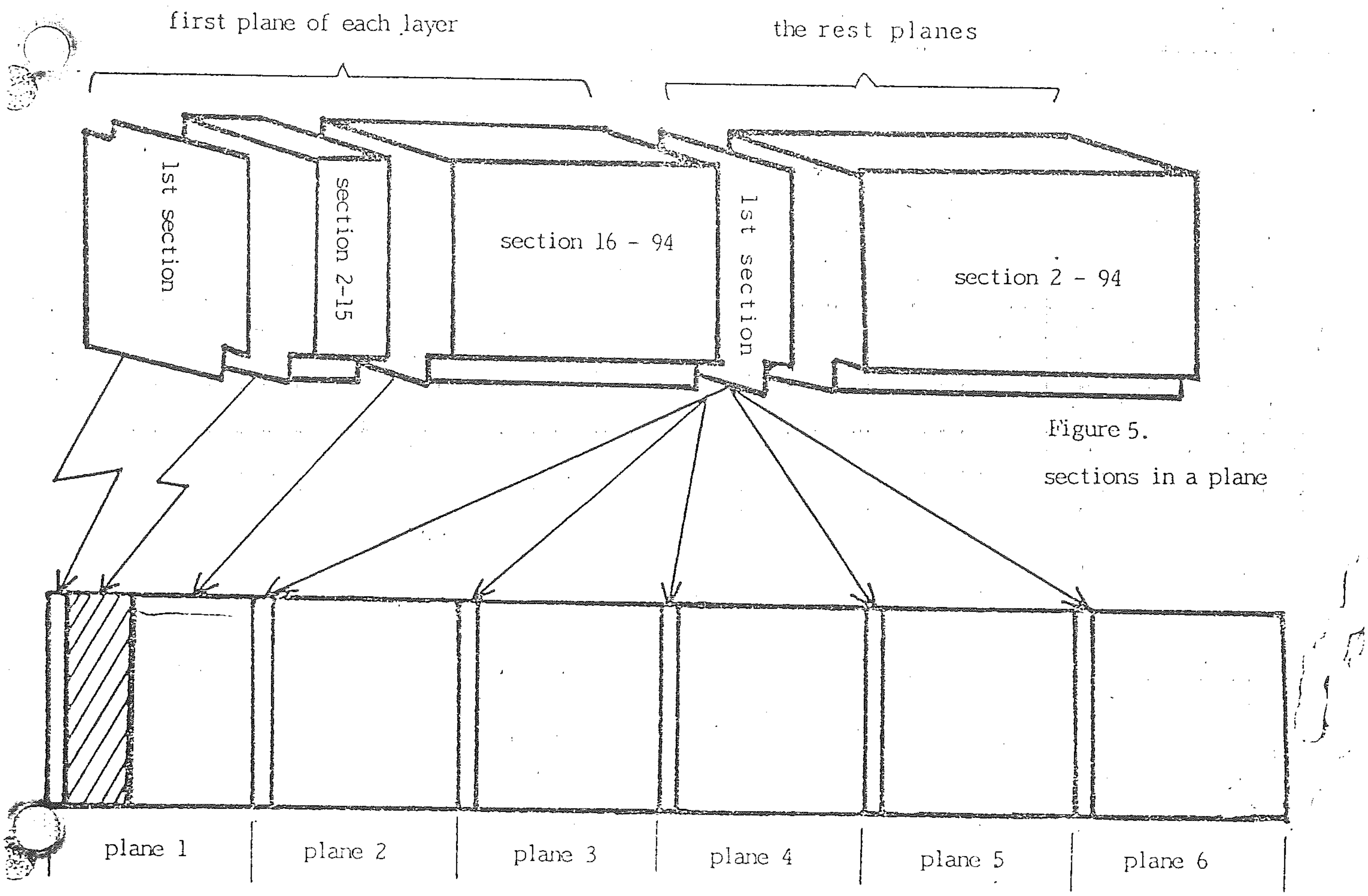


Figure 5. sections in a plane

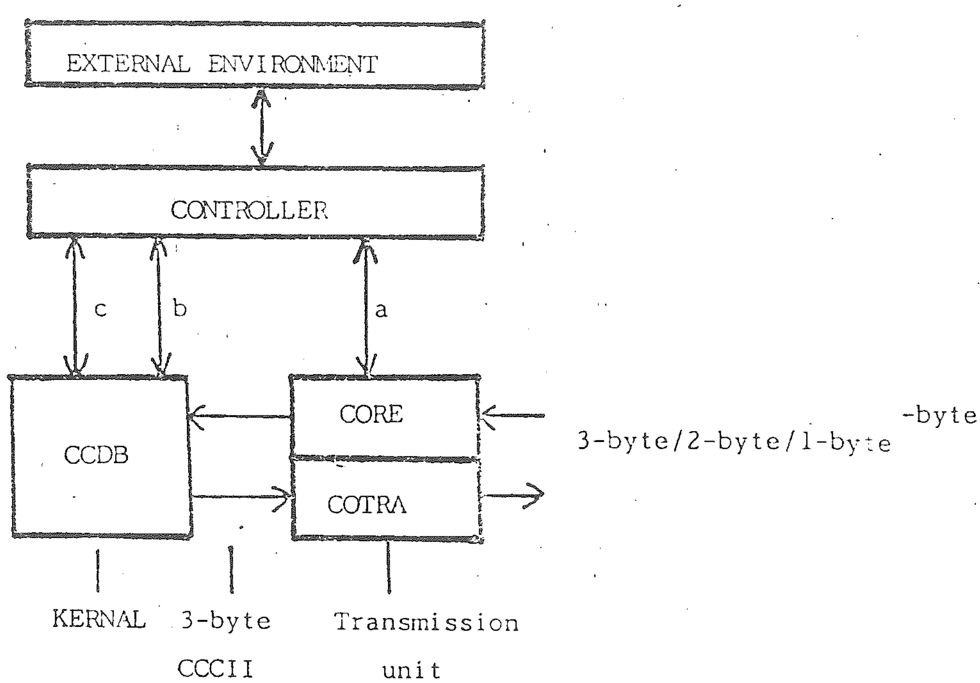
Figure 6. a layer is composed of 6 planes
the 1st section of each plane is reserved for the control codes of CCCII
section 2 - 15 of the 1st plane of layer 1 - 12 are reserved for user area

R/C	2	3	4	5	6	7
0			DWO	SCH		PSL
1			DST	SLN		PSR
2			DPR	CHS		PSU
3			DBL	CSR		SSL
4			DSE	CFS		SSU
5			DCH	CFR		
6			DPA	CDR		
7				CHL		HOM
8			DLN	HLR		
9			DPG			
10			DVO			
11			DCU			
12			DTI			
13						
14						
15						

CCCII	RAD	STK	EXP	VFM	PHC	FNT	ETL	OTC

CCCII: Chinese Character Code for Information Interchange
RAD: Radical which a character belongs to
STK: Total stroke count
EXP: Component expression of a character
VFM: Variant forms of a character
PHC: Phonetic sysbols of a character's pronounciation
FNT: Dot matrix representation of the FONT of a character
ETL: External or Indexing code of a character
OTC: Codes of different coding system

Figure 8. The Attributes Provide by CCDB



a. The routine calls from controller to CORE/COTRA and STATUS from CORE/COTRA back to controller
b. The routine calls from controller to CCDB and the STATUS from CCDB back to controller
c. Data channels

Figure 9. The organization of CCCII - MACHIME

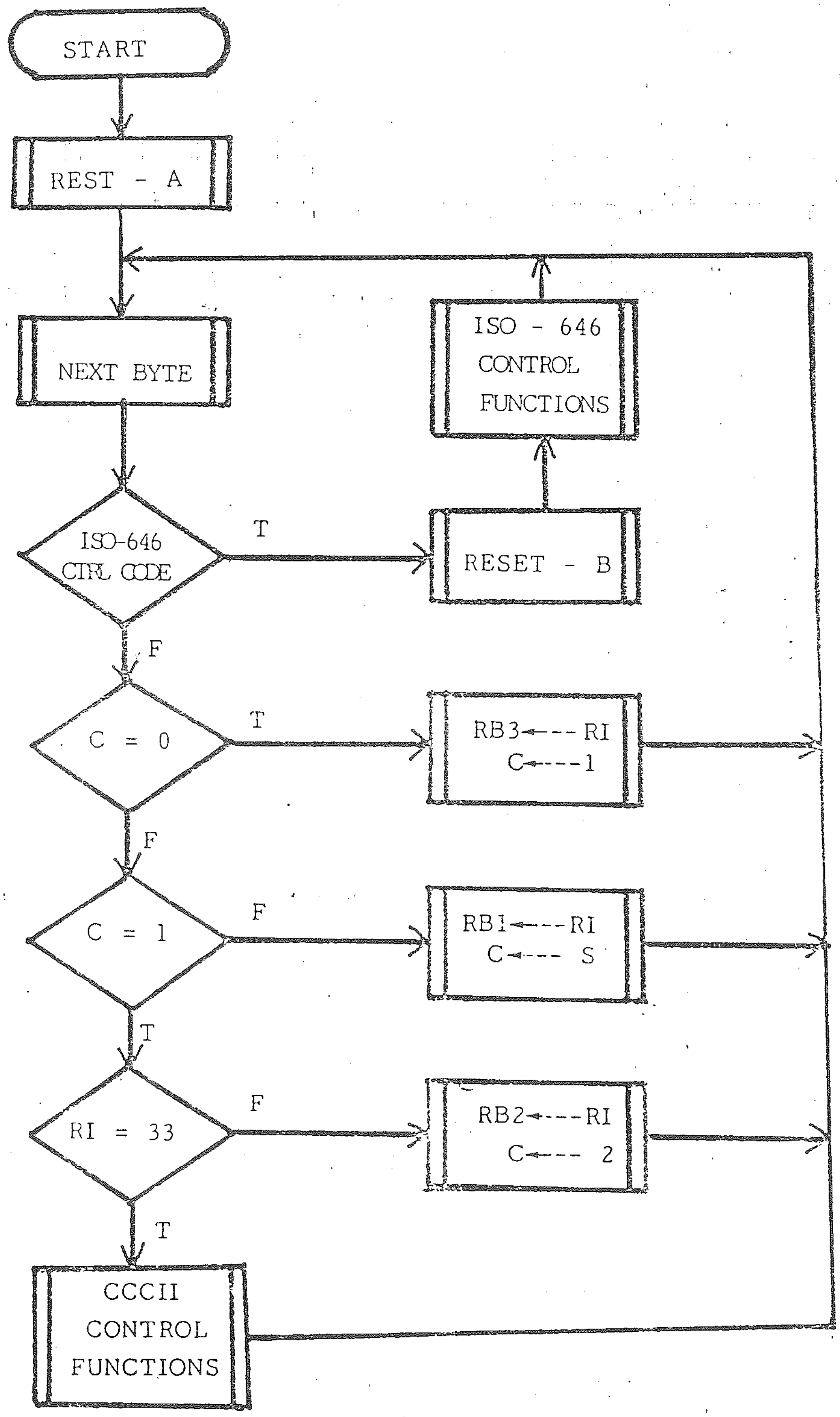


Figure 12. The main control routine of CORE

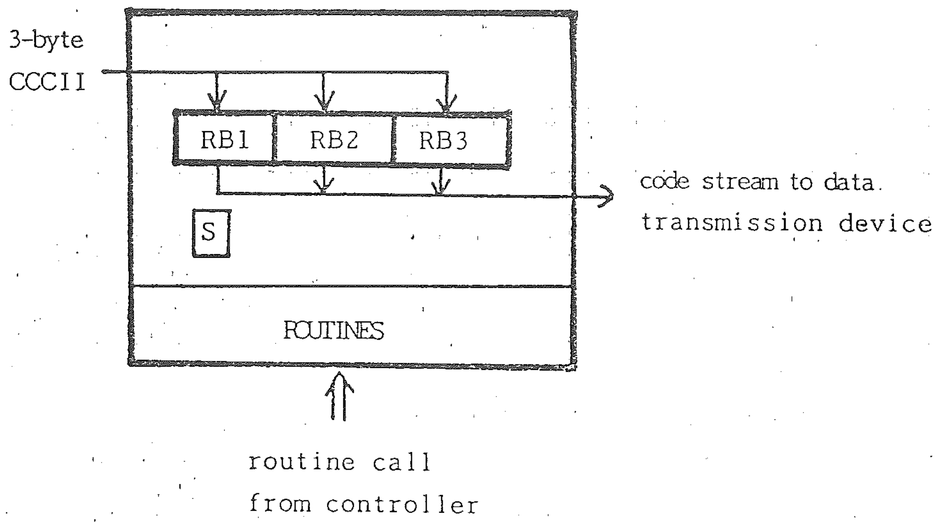


Figure 13. The structure of COTRA

S	OUTPUT	DESCRIPTION
0 ----→ 1	PSL / PSU with parameter	3-byte ----→ 2-byte
0 ---→ 2	SSL with parameter	3-byte ----→ 1-byte
1 ---→ 0	HOM	2-byte ----→ 3-byte
1 ---→ 2	SSL with parameter	2-byte ----→ 1-byte
2 ---→ 0	ESC 2/4 4/1	1-byte ----→ 3-byte

Figure 14. The relation between control code and S

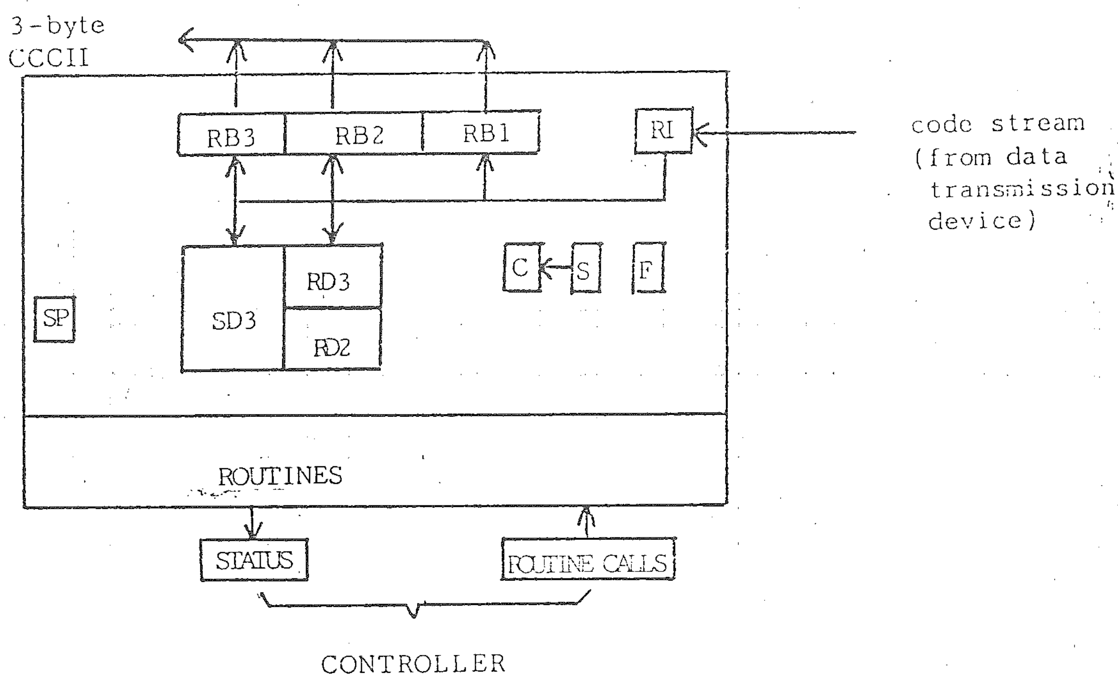


Figure 10. The Structure of CORE

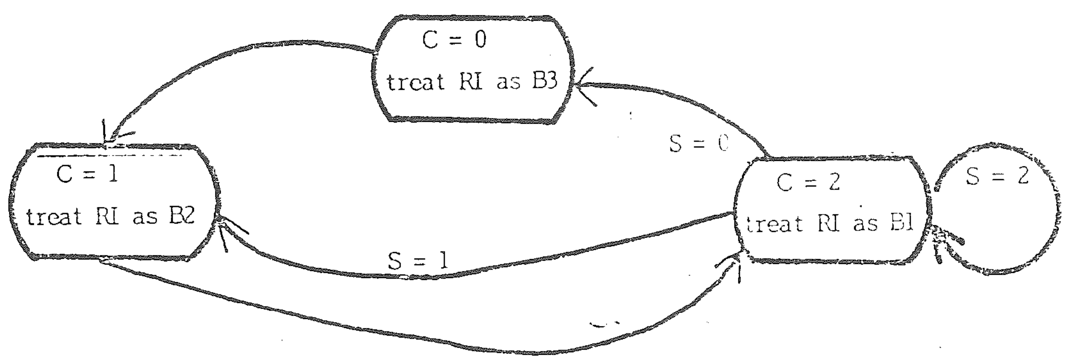


Figure 11. The functions of C and S

Reference

[1] The Chinese Character Analysis Group, "Symbol and Character Tables of Chinese Character Code for Information Interchange", May, 1983.

[2] The Chines Character Analysis Group, "The design of a cross-reference database for Chinese Character Indexing", Apr., 1980.

"Chinese Character Data Base, its Design, Implementation, and Application", at ASIS-82, October 19, 1982.

[3] The Chinese Character Analysis Group, "A Multi-lingual Coding System based on CCCII", 2nd Asian-Pacific Conference on Library Science, May 20, 1985.

[4] Karen Smith-Yoshimura & Alan Tucker, "RLIN East-Asian Character Code and RLIN CJK Thesaurus", 2nd Asian-Pacific Conference on Library Science, Seoul, Korea, May 20, 1985.

[5] The Chinese Character Analysis Group, 2nd edit., May, 1985, Taipei.

[6] C. C. Hsieh, et. al., " CCCII ", CCCII , March 13, 1986, Taipei.

Appendix A

*PSL (Plane-shift, locked) : B1=112

To set the B3 (plane) as default value, all the following CCCII's will have the same B3 and thus omitted. This will produce 2-byte (B2 and B1) CCCII's. PSL must be followed by 2-byte parameter, in which the first byte is ignored and the second byte is used as the plane indicator. The control function of PSL is explained as follow :

{[B3],B2,B1;}	[B3],33,112;	X,B3';	{B2,B1}
3-byte CCCII string or 2-byte CCCII string with old default value of B3	PSL	parameter	2-byte CCCII string with B3' default value

* PSR (Plane-shift, Locked): B1=113

Pop the last old default value of B3 from stack, if stack is bank,the PSR will be not reacted. The PSR has no parameter follows after, and it is accepted only under the case of 2-byte condition, if under the case of 3-byte condition, the PSR will be rejected.

* PSU (Plane-shipt, unlocked): B1=114

Similar to PSL. The difference is that PSL will change the B3 permanently (or until another PSL with different plane value). But the PSU will only change the following character. The PSU is accepted only under the case of 2-byte condition, if under the case of 3-byte condition, the PSU will be rejected. The function of PSU is explained as follows:

{B2,B1;}+	33,114;	X,B3';	B2,B1;	{B2,B1}+
2-byte CCCII stream with default B3+	PSU	parameter	2-byte CCCII (with B3')	2-byte CCCII stream with default B3+

* SSL (Section/Plane-shift, Locked): B1=115

Use to set the default values of B3 and B2. All the following CCCII's. SSL is applicable to 3-byte and 2-byte condition, the SSL must be followed by a 2-byte parameter, the first byte indicates the default value of B2, and the second byte indicates the default value of B1. The functions of SSL is explained as follows:

{[B2],B2,B1}	[B3],33,115;	B2',B3':	{B1}
3-byte CCCII stream or 2-byte CCCII stream with old default value of B3	SSL	parameter	1-byte CCCII stream with default B3' and default B2

* SSU (Section/Plane-shift, Unlocked): B1=116

Similar to SSL, but only react on the character that followed after. SSU only applies under 2-byte condition, SSU will be rejected if under 3-byte condition. The functions of SSU is explained as follows:

{B2,B1;}+	33,116	B2',B3'	B1;	{B2,B1;}+
2-byte CCCII stream with default B3+	SSU	parameter	1-byte CCCII (with B3' and B2'	2-byte CCCII stream with default B3+

* HOM (Home): B1=119

HOM is for deleting the defined B3 default value, to recover back to 3-byte or 2-byte , once the CCCII 3 bytes is reduced to single byte, it will need the ESC 2/4 4/1 to go back to 3 bytes codes.

Notes:

1. For all CCCII control codes, B3 is not active and B2=33.

2. All the symbol used above, their definitions are as follows:

, Used to delimit the byte, not appear in the code stream.

; Used to delimit the CCCII code, not appear in the code stream.

[ ] Used to express B3 is not exist under 2-byte condition.

{ } Used to express CCCII stream, as the number of CCCII codes = 0, 1, 2, ...;

X Used to express the byte value is 34-126.

B3' and B2' Used each expreses B3 and B2 default value respedtively.

{}+ Used to express 2-byte CCCII stream with specified default B3+.

Appendix B

```
procedure F_RESET;
  begin
    F:=NO_ERROR
  end;


procedure RESET_A;
  begin
    S:=0;
    C:=0;
    SP:=BOTTOM;
    F_RESET
  end;


procedure RESET_B;
  begin
    if C<>S then F:=IMCOMPLETE_CCCII;
    C:=S
  end;


procedure RESET_C;
  begin
    C:=S;
    F:=PARAMETER_ERROR
  end;


procedure NEXT_BYTE;
  begin
    repeat WAIT_LOOP { CORE idle }
    until EVENT(RI) { next byte into RI }
  end;


procedure ACCEPT_BYTE;
  begin
    NEXT_BYTE;
    if RI<33 or RI>126
      then F:=BYTE_ERROR
      else F:=NO_ERROR
  end;


procedure PSL;
  begin
    ACCEPT_BYTE;
    ACCEPT_BYTE;
    if F=NO_ERROR
      then
        begin
          RB3:=RI;
          SD3[SP]:=RI;
          SP:=SP+1;
          S:=1;
          C:=1
        end
      else RESET_C;
  end;


procedure PSR;
  begin
    if SP>BOTTOM
      then
        begin
          case S of
            0          :
F:=PSR_REJECT;
            1 : begin
                  SP:=SP-1;
                  RB3:=SD3[SP];
                  C:=S;
                  F_RESET
                end
          end
      else F:=STACK_EMPTY
  end;


procedure PSU;
  begin
    ACCEPT_BYTE;
```

```
ACCEPT_BYTE;
if F=NO_ERROR
  then
    begin
      RB3:=RI;
      ACCEPT_BYTE;
      if F=NO_ERROR
        then
          begin
            RB2:=RI;
            ACCEPT_BYTE;
            if F=NO_ERROR
              then
                begin
                  RB1:=RI;
                  write(RB3,RB2,RB1);
                  RB3:=SD3[SP-1]
                end
              else F:=PSU_FAIL
          end
        else F:=PSU_FAIL
    end
  else F:=PSU_FAIL;
  C:=S
end;


procedure SSL;
  begin
    ACCEPT_BYTE;
    if F=NO_ERROR
      then
        begin
          RB2:=RI;
          RD2:=RI;
          ACCEPT_BYTE;
          if F=NO_ERROR
            then
              begin
                RB3:=RI;
                RD3:=RI;
                S:=2;
                C:=2
              end
            else RESET_C
        end
      else RESET_C
  end;


procedure SSU;
  begin
    ACCEPT_BYTE;
    if F=NO_ERROR
      then
        begin
          RB2:=RI;
          ACCEPT_BYTE;
          if F:=NO_ERROR
            then
              begin
                RB3:=RI;
                ACCEPT_BYTE;
                if F=NO_ERROR
                  then
                    begin
                      RB1:=RI;
                      write(RB3,RB2,RB1);
                      RB3:=RD3;
                      RD2:=RD2
                    end
                  else F:=SSU_FAIL
              end
            else F:=SSU_FAIL
        end
      else F:=SSU_FAIL
    C:=S
  end;
```

```
procedure HOM;
  begin
    RESET_A
  end;
  begin
    ACCEPT_BYTE;
    if F=NO_ERROR
      then
        begin
          case RI of
            33 : ...... ;
                 :
                 :
            112 : PSL;
            113 : PSR;
            114 : PSU;
            115 : SSL;
            116 : SSU;
                 :
            119 : HOM;
                 :
                 :
            126 : ......
          end
  end;


procedure ESC_SEQ;
  begin
    NEXT_BYTE;
    if RI=36
      then
        begin
          NEXT_BYTE;
          if RI=65 then RESET_A
                   else ......
        end
      else ......
  end;


procedure ISO646_CONTROL;
  begin
    if RI=127
      then DELETE
      else
        begin
          case RI of
            0 : NULL;
                :
                :
            27 : ESC_SEQ;
                :
                :
            32 : SPACE
          end
  end;


procedure CORE;
  begin
    RESET_A;
    repeat
      NEXT_BYTE;
      if RI<=32 or RI=127
        then
          begin
            RESET_B;
            ISO646_CONTROL
          end
        else
          begin
            case C of
              0 : begin
                    RB3:=RI; C:=1
                  end;
              1 : if RI=33
                    then CCCII_CONTROL
                    else
                      begin
                        RB2:=RI; C:=2
                      end;
              2 : begin
                    RB1:=1; C:=S;
```

```
                write(RB3,RB2,RB1)
              end
          end
    until SYSTEM_BREAK
  end;

procedure CORE;
  begin
    RESET_A;
    repeat
      NEXT_BYTE;
      if RI<=32 or RI=127
        then
          begin
            RESET_B;
            ISO646_CONYROL
          end
        else
          begin
            case C of
              0 : begin
                    RB3:=RI; C:=1
                  end;
              1 : if RI=33
                    then CCCII_CONTROL
                    else
                      begin
                        RB2:=RI; C:=2
                      end
              2 : begin
                    RB1:=1; C:=S;
                    write(RB3,RB2,RB1)
                  end
          end
    until SYSTEM_BREAK
  end;

procedure CORTA;
  begin
    case S of
      0 : write(RB3,RB2,RB1);
      1 : write(RB2,RB1);
      2 : write(RB1)
    end
  end;

procedure OUT_LOCK_1(B3,B2);
  begig
    case S of
      0 : begin
            write(33,33,115); { SSL code }
            write(B2,B3)  { parameter }
          end;
      1 : begin
            write(33,115); { SSL code }
            write(B2,B3) { parameter }
          end
    end;
    S:=2
  end;

procedure OUT_LOCK_2(B3);
  begin
    case S of
      0 : begin
            write(33,33,112); { PSL code }
            write(34,B3); { parameter }
            S:=1
          end;
      1 : begin
            write(33,112); { PSL code }
            write(34,B3) { parameter }
          end
    end
  end;

procedure OUT_UNLOCK_1(B3,B2,B1)
  begin
    if S=1
```

```
      then
        begin
          write(33,116); { SSU code }
          write(B2,B3,B1)
        end
  end;

procedure OUT_UNLOCK_2(B3,B2,B1);
  begin
    case S of
      0 : begin
            write(33,33,114); { PSL code }
            write(34,B3,B2,B1) { parameter & 2-byte code }
          end;
      1 : begin
            write(33,114); { PSU code }
            write(34,B3,B2,B1) { parameter & 2-byte code }
          end
    end
  end;

procedure OUT_LOCK_3;
  begin
    case S of
      0 : write(33,119); { HOM code }
      1 : write(27,36,65) { ESC 2/4 4/1 }
    end
  end;

procedure PSR_CODE;
  begin
    write(33,113) { PSR code }
  end;
```