A Portable Device for Optically Recognizing Braille – Part II: Software Development

I Murray and T Dias
School of Electrical and Computer Engineering
Curtin University of Technology
Kent Street, Bentley, Western Australia 6102
Australia
i.murray@ece.curtin.edu.au | rdiast@curtin.edu.au

This article describes the software portion of the research work that developed a prototype portable device for optically scanning embossed Braille and conversion of the scanned text to binary Braille representation. The hardware development was reported in Part I of this paper [8]. An application to convert the literary Braille code to expanded text has also been implemented. The system developed utilises a hand held scanner that captures the embossed Braille image, in real time, via a linear 128-pixel CCD array. A Texas Instruments Digital Signal Processor performs recognition processing.

1 Introduction

Part one of this two-part article provides a brief introduction to the Braille system and the issues involved in optically recognizing a tactile medium [8]. In this paper a description of the software development is given.

2 Software Design

2.1 Host Computer Interface

The communications between host and DSP Starter Kit (DSK) are necessary for Braille decode from the binary Braille representation to fully expanded text. The routines to achieve this are detailed in this section. These functions (table 1) are called from Blenkhorns original code and were not significantly altered in this work [2,3].

Table 1: Significant host routines

Used In	Function Name	Description
STAND ALONE	HEXDUMP.C	Dumps the contents of a binary Braille file to screen.
STAND ALONE	TEXTTOBRL.C	Converts an ASCII file to binary Braille. Test purposes only.
CONVERT.C	CONVERT_TO_USSBC	Converts raw data to US standard Braille code
CONVERT.C	READ_LINE	Reads a line from the Braille input stream.
CONVERT.C	ADD_TO_OUTPUT	Outputs a converted character

2.2 Host and DSK Interface Routines

Texas Instruments supply a library of routines that allow communications between the DSK and a host computer [5]. The DSK and host computer communicate serially through an RS-232 cable with the host computer transmitting data from serial communications port. The host transmits and receives through its on-board UART. However, the DSK does not have a UART and therefore must simulate one. This is achieved by the use of two hardware pins known as the External Flag output (XF) and the Branch control Input (BIO). The XF line is connected to the communications port receive pin and the BIO pin is connected to communications port transmit line with the host's DTR line connected to the TMS320C50 reset pin, which remains high unless resetting the DSP [4].

An application must be executed on the host to begin DSK communications, with the program running on the host initiating all of the communication processes [4]. The communications kernel residing in the DSP responds to the hosts requests. To initiate data transfers the first word that must be sent to the DSK is a control word. After the control word has been sent, the addresses of the desired data value(s) followed by the length of the data block to transfer are transmitted. After each word has been received by the DSK, the DSK responds by sending an echo of the received data back to the host. In all communications used in this work, the echoed character is ignored [6]. Receiving data from the DSK requires reading the host RS-232 registers. However, transmission of data back to the host is accomplished by simulating an RS-232 UART as mentioned above. The data to be sent to the PC is assumed to be in the accumulator at this point.

2.3 Grade-2 Decompression

To achieve the translation of compressed Braille to expanded text, several programs were developed. They are an application to create a file to simulate the output from the scanner, a program to check the created file and finally the conversion program itself [7,8].

To simulate decoding of Braille, a utility was created to read a text file and convert the input to the format that is output by the scanner. The format from the scanner uses the high order byte of a 16 bit word with a set bit representing a dot being present and a zero as no dot present. For example, the code 10101000 in the high byte indicates that dot 1, dot 3 and dot 5 are set which is lower case o. To check the conversion is done correctly, a hex dump routine was also written that read the binary file and dumped the hex representation to screen.

The conversion routine uses much of the code developed by Blenkhorn, 1995 [2,3]. The main difference lies in the way Braille is presented to the conversion program. The original format in [3] was used to decode a file of USBC characters and create a file of ASCII text. In the prototype a single character representing dot positions is accepted by the convert application, decoded if possible, and sent to the output stream. If it is unable to be decoded, such as the case of a contraction or format sign, the code is stored and then decoded when the context has been determined. The modified algorithm for the conversion routine is detailed in Appendix A.

2.4 DSP Algorithms

The algorithms used to capture and convert the image are described here. The steps involved were specified as the real time schedule, initialisation routines, image slice capture, recognition and data transmission. The relevant routines are presented in Table 2. The flowchart depicted in figure 1 illustrates the program flow for the image capture and recognition routines.

2.5 Image Slice Capture

Illustrated in figure 2 is a CCD signal of the bright areas illuminated as the CCD first approaches the column of Braille dots (at bottom) followed by the shadow cast as the CCD and lens passes over the said column. It may be noted from this diagram that an average level is obtained at the reference points in this figure.

Table 2: Significant DSP routines

Used In	Function Name	Description
MAIN	RECOGNISE	Performs the recognition of the scan word array.
SLICERD	AICINIT	Initialise AIC on DSK board.
AICINIT	AIC_2ND	Serial port enables and sends AIC parameters.
MAIN	LOADSLICE	Reads one CCD sample and determines dot positions.
MAIN	RECOGNISE	Performs correlation of Cell maps to scan word array.

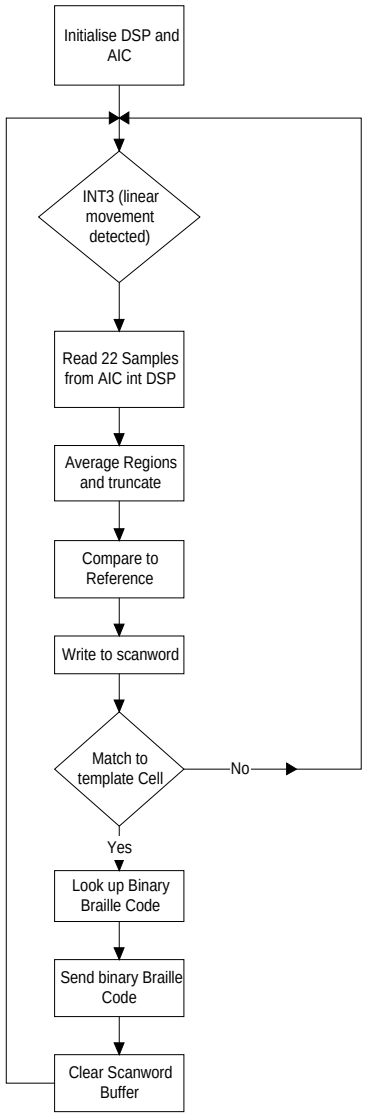

Figure 1: Program flow chart

One major concern in this work was the fact that Braille is embossed on many forms of material, i.e. high quality paper, plastic thermoform (beige coloured) and standard 80 GSM paper. CCD voltages obtained from these mediums differ quite markedly. By obtaining an averaged reference level from areas that should contain only reflection from blank paper (2), automatic con-

trast, with respect to the backing medium, may be obtained. This will then yield a reference level to threshold the incoming signal.

From the fact samples are captured, from the time of the SI pulse, at 100µs intervals,[6] samples 1 through 4 may be discarded as they lie off the lens area. The same applies to samples over number 22. Samples 5, 6 and 20,21 should, if the device is correctly aligned, provide the reference level. This has the added advantage that as the reference level is obtained for every slice taken, dirt appearing in the image should, in all but the most extreme cases, be compensated for. These samples, when averaged, also overcome the discrepancy in CCD array sections. Dot positions are obtained by averaging and saving the relevant samples for those positions. Figure 3 illustrates a captured slice over the illuminated section of the Braille cell. There appears in the image a step where the second 64 pixel array commences.

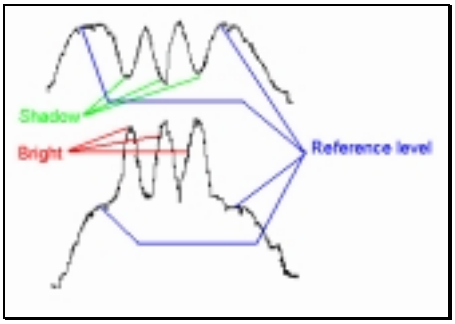

Figure 2: Image slice signals

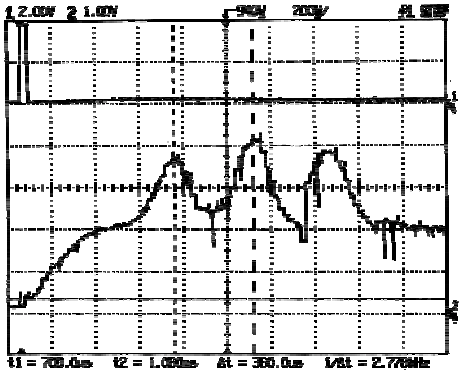

Figure 3: Si pulse and the image of three raised dots

It is worth noting the overlap as an attempt to overcome positional errors. This method is inferior to the application of fuzzy logic. By the use of membership functions, a far more intelligent decision as to dot location may be made. This concept will be marked for further investigation.

It must be mentioned at this point that for interpoint Braille to be read, it is necessary to know the order of bright and shadow. If an interpoint dot is scanned, the bright/dark order is reversed. As the interpoint dot is a dip in the paper, CCD signals showed that the dot appeared as a shadow followed by a bright (low CCD level followed by high CCD level). In the context of Braille OCR this is an important point of consideration.

Once the image of each vertical scan is captured, averaged and compared to the reference level they are stored as depicted in table 3.

Table 3: Dot position within the scan word

Sample Number	Dot Position
7,8,9,10,11	dot one
11,12,13,14,15	dot two
14,15,16,17,18,19	dot three

Vertical slices are stored in a single byte, position oriented (table 4). That is 3 pairs of 2 bits representing whether that position is bright, shadow or reference. Two bits are reserved for either 8 dot Braille or status. These results are captured in a first in first out (FIFO) buffer of 50 vertical slices, or scanwords. As slices come into this array, the old slices are rolled off the storage area and the new slice is placed on the top of the stack. The circular buffer used in conjunction with the data move abilities of the TMS320C50 allows the FIFO buffer to be maintained with very few overheads. The code fragment detailed below illustrates that just 2 instructions are required to maintain the FIFO buffer and safeguard against overflow and underflow of that buffer.

After each slice capture an attempt to recognise the image in the FIFO buffer is made. This is achieved by a bit mask comparison with ideal masks

Table 4 : Scan word storage

Bright	Shadow	Reference	Status
10	01	00	XX

On a successful match, the binary Braille code is transmitted to the host for decoding into expanded text and the buffer is cleared. Otherwise the system waits for the next vertical slice to see if that completes the cell. With this method 50 slices equates to 6.35mm and as Braille cells are 4mm edge to edge allows a slight overlap.

3 Device Testing

3.1 Image Capture and Scanner Testing

For testing purposes, test patterns of Braille dot combinations were produced. These patterns are ideals, in that the embossing is not degraded by use and they are produced on a Perkins Brailler in good condition, yielding high quality dots.

All alignment of scanner and illumination as well as parts of the recognition process was performed using these ideal sheets. To simulate marks, ink and pencils were used. "Flattening" a test sheet, achieved by placing it under several books, created worn Braille. This last aspect closely resembles the way stored Braille degrades. The overall image capture was clear and well defined. Adequate voltage levels for comparisons were achieved even on worn and thermoform Braille.

3.2 Recognition Algorithm Testing

For testing purposes, two reference cells were developed for the correlation routine matching the incoming scan words. The first had all dots set, that is dots 1,2,3,4,5,6 are raised, and the second with dots 1,4,5,6 raised. This was considered sufficient for testing and evaluation of the recognition algorithm.

3.3 Braille Decode Software Testing

Several applications, as described previously, were constructed to simulate the Braille decode section of this project. Under the test conditions a limited number of exceptions were incorporated. It was not expected that all contractions will decode correctly, in particular the syllable boundary rule. Simulations indicated that on implemented rules and contractions, the convert program functioned as intended.

4 Conclusions

4.1 Achievements

A device that optically scanned Braille and recognised the cell was successfully developed. The scanner assembly provided signals of a suitable quality and level to enable processing of the relative dot positions with a high degree of reliability and flexibility. The process of decoding literary Braille [1] to text was achieved, although not all contractions in English Braille were implemented. This is of little consequence, as alternate rules will be required in the near future when the new standard for universal Braille code is released.

Partial implementation of the cell recognition was completed. This section requires storage of cell patterns to correlate with the stored data in "scan word". This was a major task in itself and was not given a high priority, as proving the approach required only some sample cell maps. Additionally, the cell maps are dependant on the linear rate of movement or slice captures per inch. As it is intended to alter this specification in future work (detailed in section 4.3), few cell maps were generated. The storage of these maps had an influence on the slice rate capture, at 200 slice captures per inch, 2048 words are needed for storage. This reduces to 756 words at 75 slice captures per inch.

4.2 Significant Achievements

The most significant achievements of this work can be separated into two areas. They are image capture of a tactile medium and the digital signal processing recognition system.

As stated earlier, the capture of a tactile written medium has unique demands. The method of capturing the image of an embossed cell by selfoc lens, oblique illumination and linear CCD array provided excellent imaging and allowed for a much reduced level of processing when compared to area arrays and commercial flatbed scanners. It must be stated, however that this style of scanner has a major disadvantage. It is difficult to keep vertical with respect to the Braille line and tends to wander off the ideal alignment.

4.3 Recommendations for Future Work

There are several aspects of the prototype that require further development to bring this device to a point that it is of practical use.

4.3.1 Storage of Cell Maps

For testing purposes, each cell map contains 32 words, covering column one, the column break and column two. This method was simple to implement but also inefficient. The most obvious inefficiency is in the column break. This area should contain all zeros, i.e. at reference level, for sixteen samples (scan words). Therefore there is no need to store a zero array. A check to see if sixteen consecutive words are all zero would perform the same function. As the pattern necessary for each column combination is 2^3, or eight possible variants, and column one possibilities are

identical to column two possible dot patterns, there is no need to store both. A check of column one or two will yield an octal number for that column. The result being a two digit octal number corresponds to the unique dot pattern of that cell. Some examples are shown in Table 5.

Table 5: Improved cell map

Column one (Dots set)	Column two (Dots set)	Result Octal Number
---	5--	04
1--	---	40
--3	--6	11
---	4,5,6	07
1,2,3	---	70
1,2,3	4,5,6	77

A look up table using the octal reference may be used to transmit the assigned binary Braille code to the host.

By this method, cell maps would be an array of eight words. Just seven of each would be required, as a cleared column is not required to be stored, i.e. one that has no dots set could be checked for zero contents as described for column spacing. This requires 64 words of cell map storage compared to 2048 words as implemented in the prototype.

4.3.2 Recognition of Dot Position

The use of fuzzy logic in determining dot position may allow for positional errors within each vertical slice better than a crisp decision as used in the prototype. The dot position could be determined by the strength of its membership in a fuzzy set, thereby allowing a greater degree of positional flexibility.

5 References

[1] Bledsoe W, "Braille: A Success Story, *Evaluation of Sensory Aids for the Visually Handicapped*", National Academy of Sciences, Washington U.S.A. pp 3-36, 1972

[2] Blenkhorn P.L, Personal Transcription Systems, Computerised Braille Production, *The Proceedings of the 5th Annual Workshop,* Winterthur October 30-November 1,1985

[3] Blenkhorn P L, "A system for converting Braille into print", *IEEE Transactions on Rehabilitation Engineering,* vol. 3 no. 2, pp. 215-221, 1995

[4] Texas Instruments, *TMS320 DSP Designers Notebook,* Volume one, Texas Instruments, USA, 1996.

[5] Texas Instruments, *TMS320C5X DSP Starters Kit Users Guide,* Texas Instruments, USA, 1996

[6] Texas Instruments, *TMS320C5X Users Guide,* Texas Instruments, USA, 1997

[7] Murray, I, *A Portable Device for the Recognition of Braille,* Thesis, School of Electrical and Computer Engineering, Curtin University of Technology, 1998

[8] Murray, I. and Dias T, *A portable device for Optically Recognizing Braille – Part I: Hardware development*" submitted to ANZIIS 2001

Appendix A : Modified algorithm for the conversion routine

```
initialise and load tables
while not end of input do
begin
 input text is read in
 while still converting do
  begin
  start at rule defined by current character
  match = FALSE
  do
   if focus matches
       and state is ok
       and right context is ok
       and left context is ok
    begin
    output right hand side of rule
    set new state
    match = TRUE
    move along input buffer
    end
   else go to next rule
  until match
  end
end
```