AN INTERACTIVE COMPUTER GRAPHICS FACILITY USING A LOW SPEED DIGITAL COMPUTER

P.A.V. Thomas, Ph.D.
University of Windsor

1. Introduction

During the last two years, development has taken place to construct a relatively inexpensive interactive computer graphics facility about an existing PDP 8/S digital computer. Because of its low speed of operation, particularly with regard to input-output transfers, a number of interesting problems were posed and have been largely overcome. In the past, this problem of providing a flicker-free display with low speed transfer of data has been overcome by using a storage type of cathode ray tube but this is not convenient if the display is to be interactive; keyboards have been used and some work is being carried out using "write through" techniques. In the present case it was decided to use a light pen for interactive work together with a non-storage cathode ray tube and devise means to provide the required data at the speed required. Essentially two mechanisms were utilised to achieve this result: (1) keep the amount of data to a minimum and (2) provide a means of supplying this data at the rate that is required. The former has been achieved by providing hardware generators for the various data forms and the latter by providing a suitable refresh memory buffer and controller between the digital computer and the generators.

2. Data Form

Before considering the details of the system it is necessary to describe the form of the data used in the display. Essentially this can be said to consist of three types: (i) data applying to a particular item, be it alphanumeric character or a line segment, and referred to as an Item data, (ii) data applying to a particular mode which energises a particular graphical generator and referred to as a Mode data, and (iii) a Jump instruction (Jump data) which permits a particular display file to be refreshed or entry made to a new file. There are a number of methods used to organize such a display file and due to the short word length of the particular computer (12 bits), these three types of data each occupy one word in a display file. This naturally leads to three different types of data format which are shown in an Appendix in which it will be observed that the type of data is identified by the first two bits of the word which assists in the decoding by the Refresh Memory Controller to be discussed later; the remaining bits of information specify the particulars pertinent to the particular data item.

3. Refresh Buffer Memory

The primary purpose of this buffer is to match the speeds of the input/output (I/O) transfer of data through the computer (approximately 4000 words/sec) to the graphic item generation (1μs to 40μs). Various forms of buffer storage were considered, that is, drums, discs, delay lines, and core storage with respect to speed and the required capacity which should not be less than 2000 words. The ultimate decision was to use a core memory of 4096 words similar to that used in the computer. This not only gives adequate space but also eased the design.

Associated with this memory there are two single word buffer registers and two addressing registers for loading of data into the memory from the computer and accessing the data to the various graphic generators. Furthermore having accessed the data from the memory it is necessary to have a Controller capable of interrogating the data and determining the required action. The complete system, shown in Figure 1 has been fully described elsewhere(1) and will be only briefly described here.

When initially turned on the system is static. A computer override command will normally be given from the computer, through a device selector to the Data Flow Controller(DFC), to give the computer complete control of the memory. Under these circumstances the normal sequence is to first load into the memory I/O Address Register (IOAR) the location of the first data item to be loaded using a Memory Address Load command. Following this an Input Load command is given which transfers data from the computer to the I/O Buffer (IOB) and then triggers a write cycle of the memory to transfer the data from IOB into the memory location specified in IOAR. This process may be repeated as often as required; however in practice if a file is being loaded all data will occupy sequential locations so that automatic incrementation of IOAR may be initiated using an automatic increment IOAR command thereby saving the time of continually loading IOAR.

When a file is completely loaded and a display is required, the computer override is turned off which automatically starts the refresh side of the memory with the Display Address Register (DAR), and Buffer Register (DBR) cleared. The first read cycle from the memory therefore automatically starts from location 0 and transfers the contents of this location to DBR. Typically this first data item will be a Mode data and will be decoded as such due to bits 0-1 being 10. This will turn on the appropriate Generator (A/N, Vector or Dot, to be described later) and set condition flip flops particular to that generator. At the same time the DAR is automatically incremented. The next item of data is then immediately read into DBR and in this case will normally be an Item data which is detected as such by bit 0 being a '0'. The active generator is then checked for readiness to accept data whence the data is transferred to its own buffer register; again DAR is incremented and a memory read initiated. Further Item data are displayed until another Mode data is detected when either the condition flip flops in the active generator may be changed or the active generator is turned off and another turned on. This process continues until a Jump data is detected in DBR by bits 0-1 being 11. In this case bit 2 is checked for a 1 (indirect jump) 0 (direct jump); in the former case the next memory location is read into DBR and transfered directly into DAR setting the location for the next item. In the second case the automatic incrementation of DAR is inhibited and only bits 4-11 of DBR are transfered into DAR permitting a jump to the current page, assuming that bit 3 is 1; if bit 3 is 0 (indicating page Zero-the first 256 words of the memory) bits 0-3 of DAR are cleared. The reason for having the two types of Jump data is that the direct jump is limited to 512 words whereas a typical file may be longer. The indirect jump permits jumping to any location but takes two memory cycles. The use of a page zero also permits a jump to be carried out in one memory cycle to access the first item of the file (at location 0) for refreshing purposes.

If now the file has to be modified, control can be returned to the computer by means of the computer override feature mentioned earlier. Alternatively "cycle stealing" may be used by giving an Input Load Command which loads IOB and signals a Request to DFC. As the refresh side is asynchronous and data generation may take a greater time than a memory cycle (8 μs), further data cannot be supplied to the generator buffer: the I/O Request is now detected and the contents of IOB written into memory location specified by the contents of IOAR.

Having described the methods of memory access by the computer and display tube, it is appropriate at this point to discuss the method of interaction using a light pen. A light pen register (LPR) normally tracks the current value in DAR; if however a light impulse is received by the light pen and the particular item being displayed is light pen sensitive (determined by bit 9 of the last Mode word) a signal is supplied to DFC which generates a programme interrupt to the computer and inhibits any further change in LPR, whilst allowing the display to continue. In response to the interrupt the computer can then read the contents of LPR through IOB; this identifies the item "picked" by the light pen, and the user can make any required changes through the computer. After servicing the interrupt the LPR is reactivated so that it continues tracking DAR.

4. Alphanumeric Character Generator

The alphanumeric character generator is a 5 x 7 dot matrix generator that was implemented entirely with NAND logic and described elsewhere[2,3]. Briefly, the matrix is formed using Y- and X-up-down counter registers feeding digital/analogue converters (DACs) the former being incremented up to 7 by the main clock and the latter being incremented to 5 on each 7th pulse and a further double count after this to provide spacing between characters. At the same time a brightening pulse train is generated according to the character to be displayed, the basic logic being shown in Figure 2. The clock pulses are counted up to 35 in a 6-bit Dot counter, the outputs of which feed a Dot Decoder consisting of 35 7-input AND gates which provide 35 discrete 'dot' pulses #1 - #35 in time sequence. These pulses are then fed into a Character Encoder which consists of a set of OR gates whose inputs (3-17 in number) consist of the required Dot pulses; thus there are 60 'character' outputs from this selector corresponding to the 60 character ASCII code set. Prior to the clock being initiated, the 7-bit ASCII code has been loaded into a Character Buffer; the least 6 bits are then fed into the Character Selector which consists of 60 7-bit AND gates, each gate having as input the 6-bit ASCII code and corresponding 'character' train. The outputs of these gates are then ORed together and fed to the brightening circuitry. Having described the basic principles used some specific details will now be given. Three sizes of character are available, the size being controlled by altering the counting position of the 10-bit X- and Y-counter registers. Basically this permits a display area of 1024 x 1024 raster units; in the case of character generation the smallest size uses increments of 2 raster units/dot position of the matrix but by altering the counting to the 4 and 8 bit positions the character size is doubled or quadrupled permitting 3 character sizes under programme control. Furthermore a small character normally considered as a subscript may be moved up one character height for superscripting purposes. Due to this method of generation it should be pointed out that there is a minor limitation in that a character can only be positioned at discrete positions, being a multiple of one character size.

In fact the Character Buffer is 7 bits in length whereas only 6 bits are used for the character selection. The reason for the 7th bit is to allow for certain special characters, back space, line feed, character return (corresponding to carriage return on the teletypewriter) and tabulation which have similar 6-bit codes as some of the alphanumeric characters. In these cases all 7 bits are decoded and if a special character is decoded, character generation is inhibited. The operations performed are self explanatory and will be only briefly mentioned. Line feed may be single or double depending on the value of bit 4 in the data word which controls the decrementing of the Y-counter register. In the case of character return the return is again controlled by bit 4, being to the left hand margin (by clearing the X-counter register to zero) or to one of 8 positions equally spaced across the screen depending on the value (0-7) previously set in a special 3-bit Margin Register. This register is set by a Data word (whilst in the alphanumeric mode) having the desired number in its bits 2-4, the remaining bits being zero corresponding to a 'Null' character which is also a special character and detected as such to inhibit any generation. Tabulation may also be performed and automatically places the following character at the next margin position.

5. Dot Generator

It is intended to provide two methods of line drawing, these being a Vector Generator and a Dot Generator, the former being relatively slow compared to the latter, but more flexible in line positioning. The dot generator, as constructed permits a dot to be generated after incrementation and/or decrementation in both the X and Y direction by 0, 1, 2 or 4 raster units, this being achieved by counting at the appropriate point of the X- andY-up-down counter registers. Furthermore a series of equally spaced dots up to a maximum of 15 in the same direction may be generated by loading a repetition counter, within the generator, with the number specified in bits 2-5 in the data word.

In addition, a vector generator using analogue integrators is under construction which will generate lines of any length and at any angle. At this stage only point or positioning is possible in this mode, this being carried out by loading the required absolute values of X and Y into the corresponding counter registers and applying a brightening pulse after the DACs have settled in the case of the point mode.

6. Brightness Control

Three basic levels of brightness are available, those being set by bits 10-11 of the Mode words. The brightness level is set up by controlling amplitude rather than the width of the brightening pulses mentioned in the generators, 3 electronic switches picking off the appropriate potentiometer tapping; furthermore when the vector generator has been completed a particular line may be blanked out by bit 1 of the X-data word to save unnecessary vector mode words. A further feature was required when in the alphanumeric mode to compensate for the character size. When changing character size the apparent brightness of a character increases as the size is reduced due to the dots being closer together; to compensate for this 3 more electronic switches and potentiometers are incorporated, the former being under the control of the size bits of the Mode and Data words in this case. One other factor that effects the brightness of the display is the length of the file which causes a variation of the refresh rate, the rate decreasing as the file lengthens. To overcome this the file length may be kept constant by filling the unused portion with Null characters whilst in the alphanumeric mode. This, however, is inconvenient and wasteful of buffer memory space, so that a frame synchronisation feature is made available in the vector mode word (bit 6); by using a single flip flop which is turned on by sensing this bit as '1' and off by a 30Hz pulse generated from the 60Hz power supply, a picture frame is started every 1/30 second irrespective of the length of the file, this being the longest practical time to prevent flicker of the display.

The only other feature provided is provision for 'blinking' any Data item by setting bit 1 of the data word. In this case the effective refresh rate for these data items is reduced to 3-3/4 Hz which gives a satisfactory flicker.

References

(1) "A Data Flow Controller and Refresh Memory for a Computer Display System", D.L. Murray, MASc. Thesis (Windsor 1970).

(2) "An Alphanumeric Text Generator for a Computer Display", W.E. Mennie, MASc. Thesis (Windsor 1970).

(3) "A Logic Character Generator for Use in a Computer Graphics Facility", P.A.V. Thomas and W.E. Mennie, Proc. 4th Can. DECUS Symposium (1970).

Acknowledgement

The author wishes to thank the National Research Council for financial assistance to this project. Also to Messers. W. Bristow, W.E. Mennie and D.L. Murray without whom this work would not have yet been carried out.

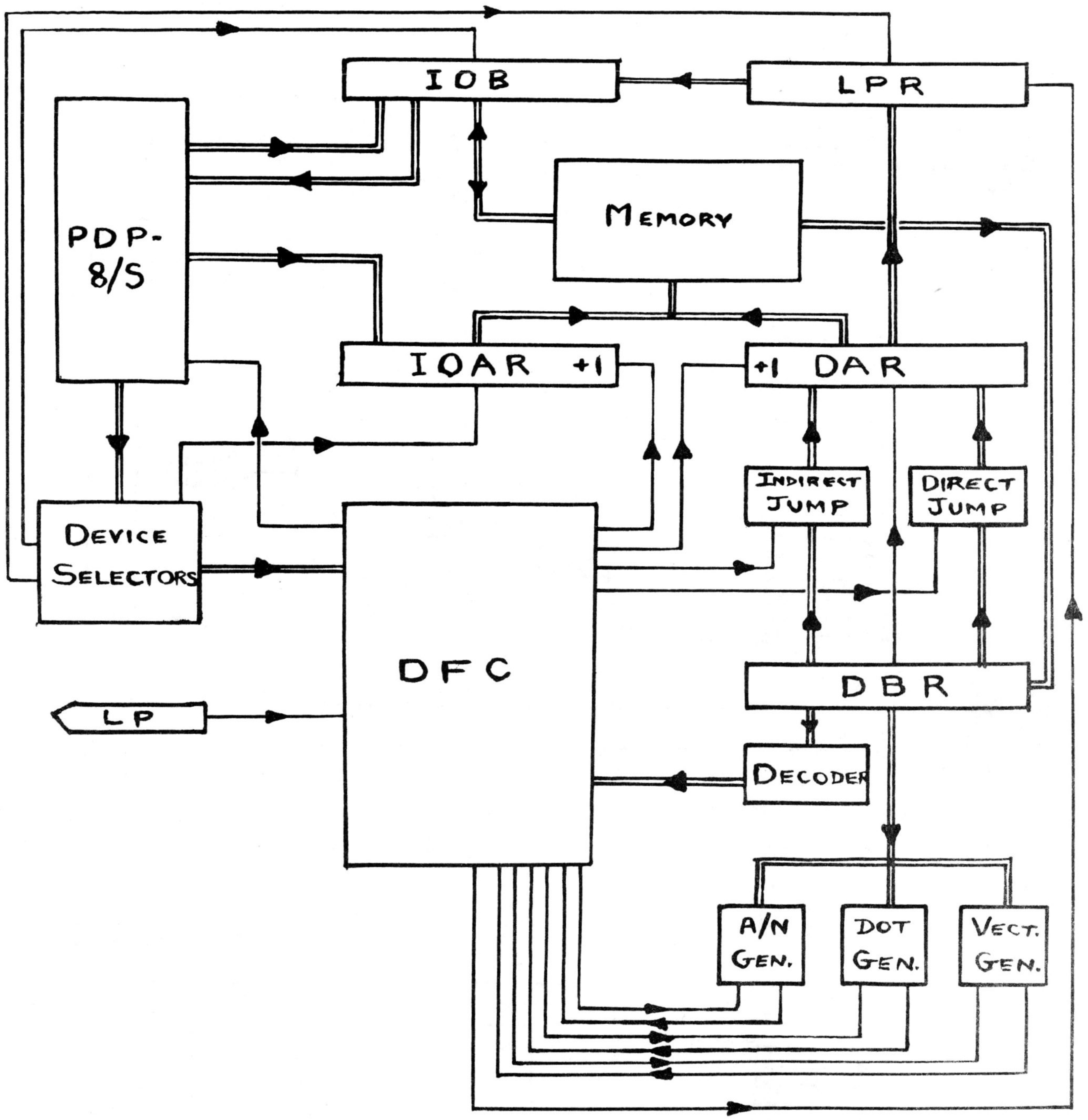

Figure 1 Overall System

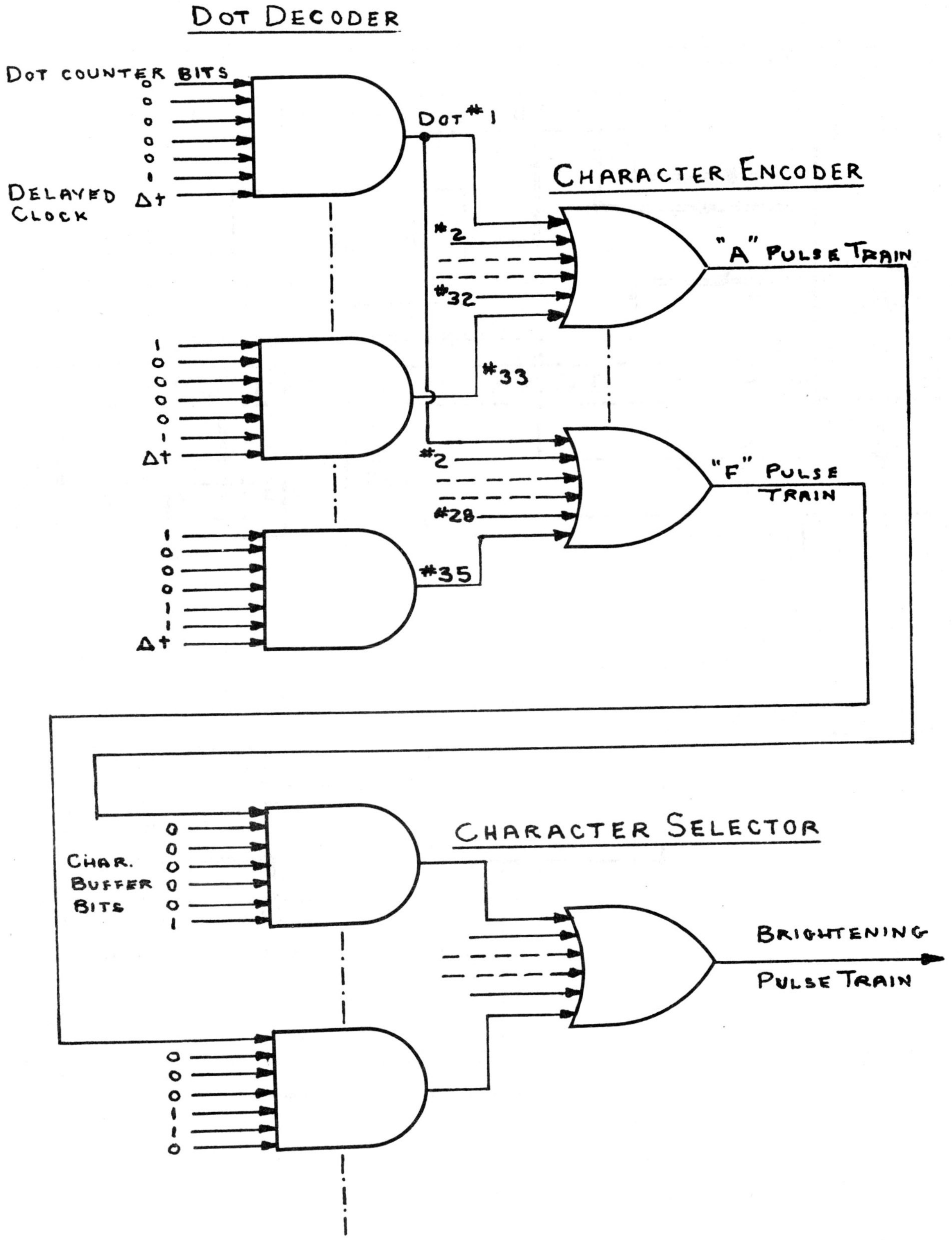

FIGURE 2 A/N PULSE GENERATION

APPENDIX

Data Forms

MODE DATA

Vector Mode. Bit positions

0	1 2 3 4	5	6	7 8	9	10 11
1	0 0 0 0		F	T	L	I

Dot Mode.

1	0 0 0 1		L	I

Alphanumeric Mode.

1	0 0 1 0	M		S	L	I

JUMP DATA

Direct Jump.

1	1 0	P	Core Location

Indirect Jump.

1	1 1	

ITEM DATA

Vector Mode. 2 words

0	Z	X co-ordinate
0	B	Y co-ordinate

Dot Mode.

0	B	R	Δ X	Δ Y

Alphanumeric Mode.

0	B	SS	M	7-bit ASCII

Margin Load.

0		MS	0 0 0 0 0 0 0

Symbols used

F :	0 - No frame synch.
	1 - Frame synch. on
T :	00 - Position only
	01 - Dot at tip
	10 - Dashed line
	11 - Solid line
L :	0 - Light pen insensitive
	1 - Light pen sensitive
I :	00 - Normal
	01 - Dim
	10 - Bright
M :	0 - Normal
	1 - Load next word into Margin register
S :	0 - Normal size
	1 - Double size
P :	0 - Page Zero
	1 - Current page
Z :	0 - Normal
	1 - Blank (overrides I)
B :	0 - Normal
	1 - Blink
R :	1 - 15 - No. of repeats
SS :	00 - Normal
	01 - Superscript
	10 - Subscript
	11 - Capital
M(CR) :	0 - Normal
	1 - Column return
M(LF) :	0 -Double space
	1 - Single space
MS :	0 - 1st.position
	to
	7 - 8th.position