## 1.1. MCU Interface

MCU Interface consist of 8 data pin and 5 control pins. The pin assignment is summarized in Table 1-1.

**Table 1-1 : MCU interface assignment under different bus interface mode**

| Pins | Data/Command Interface | | | | | | | | Control Signal | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Interface | **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** | **E** | **R/W#** | **CS#** | **D/C#** | **WAIT** |
| 8b/6800 | D[7:0] | | | | | | | | E | R/W# | CS# | D/C# | wait |

### 1.1.1 MCU Paralell 6800-series Interface

The parallel interface consists of 8 bi-directional data pins (D[7:0]), R/W#, D/C#, E and CS#. A LOW in R/W# indicates WRITE operation and HIGH in R/W# indicates READ operation. A LOW in D/C# indicates COMMAND read/write and HIGH in D/C# indicates DATA read/write. The E input serves as data latch signal while CS# is LOW. Data is latched at the falling edge of E signal.

**Table 1-2 : Control pins of 6800 interface**

| **Function** | **E** | **R/W#** | **CS#** | **D/C#** | **WAIT** |
|---|---|---|---|---|---|
| Write command | ↓ | L | L | L | L |
| Read status | ↓ | H | L | L | L |
| Write data | ↓ | L | L | H | L |
| Read data | ↓ | H | L | H | L |

**Note**

(1) ↓ stands for falling edge of signal
(2) H stands for HIGH in signal
(3) L stands for LOW in signal

**Figure 1-1 : Set Column & Row Address procedure – set a start- and end address of column & row**

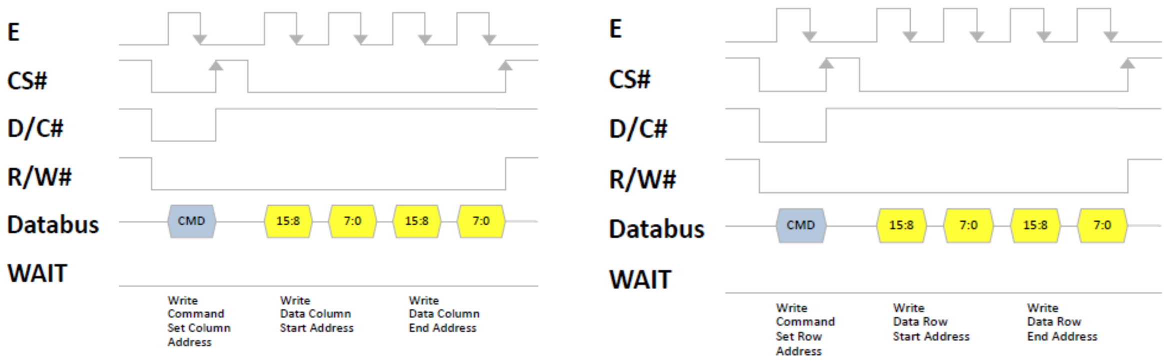
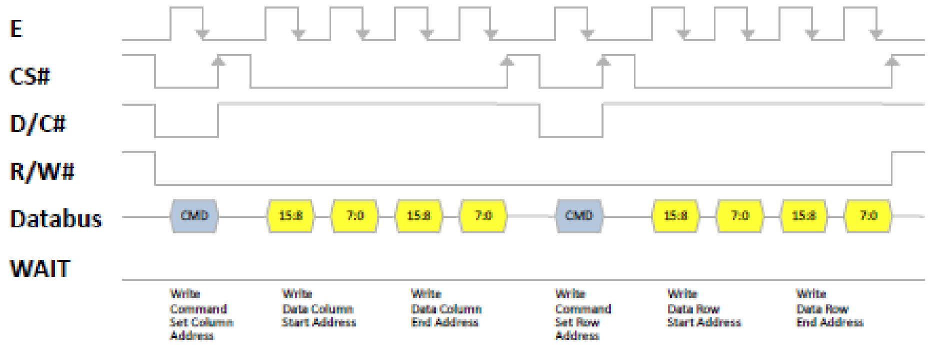

E
CS#
D/C#
R/W#
Databus
WAIT
CMD
15:8
7:0
15:8
7:0
Write Command Set Column Address
Write Data Column Start Address
Write Data Column End Address
E
CS#
D/C#
R/W#
Databus
WAIT
CMD
15:8
7:0
15:8
7:0
Write Command Set Row Address
Write Data Row Start Address
Write Data Row End Address

**Figure 1-2 : Set Column & Row Address in one flow procedure**


E
CS#
D/C#
R/W#
Databus
WAIT
CMD
15:8
7:0
15:8
7:0
CMD
15:8
7:0
15:8
7:0
Write Command Set Column Address
Write Data Column Start Address
Write Data Column End Address
Write Command Set Row Address
Write Data Row Start Address
Write Data Row End Address

In order to match the internally operating frequency of display RAM with that of the microprocessor, some waiting of extern read processing is internally performed which requires the insertion of a high at WAIT pin before the first actual display data can read. This is shown in Figure 1-3.

**Figure 1-3 : Data read back procedure - put a High of WAIT pin if VRAM busy for read**

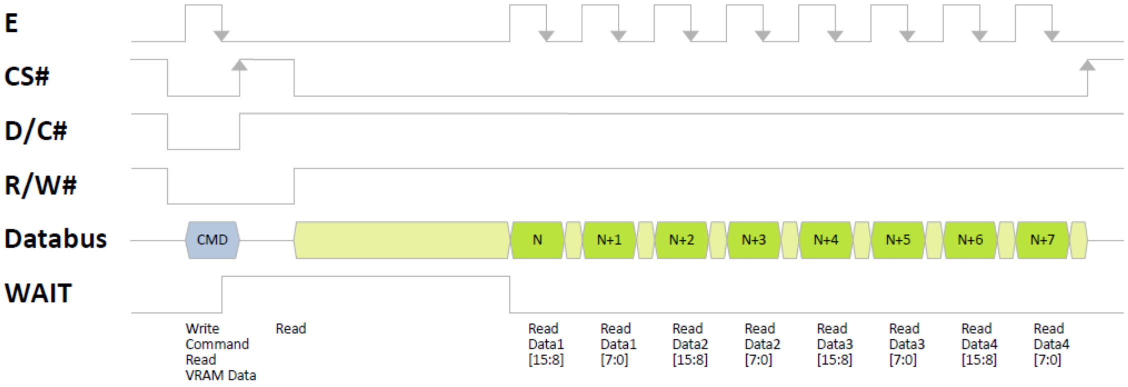
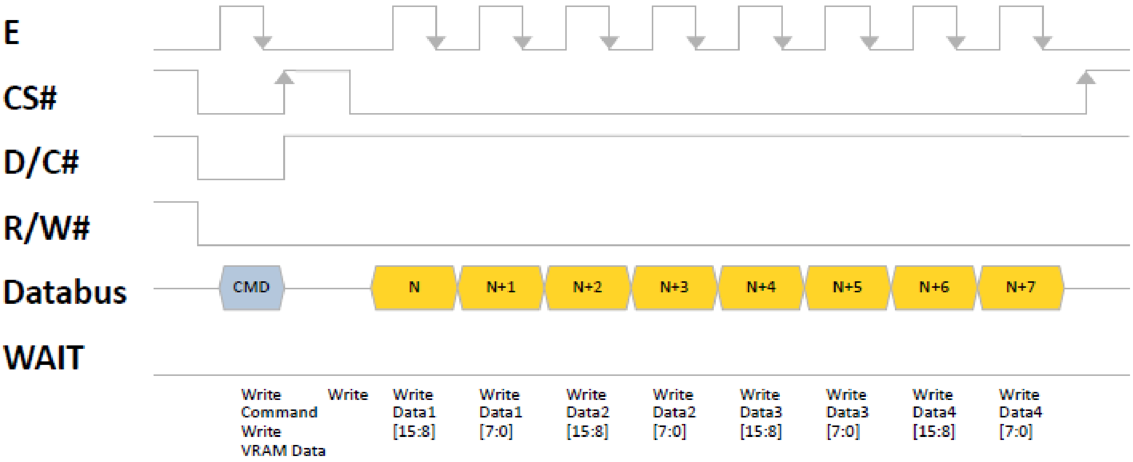


**Figure 1-4 : Data write procedure**



### 1.1.2 Databus to VRAM mapping

**Table 1-3 : Data bus to VRAM mapping**

| Write Data | | | Data Bus | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Bus Width | Color Depth | Input Order | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| 8 bits | 65K | 1st | B4 | B3 | B2 | B1 | B0 | G5 | G4 | G3 |
| | | 2nd | G2 | G1 | G0 | R4 | R3 | R2 | R1 | R0 |

## 2.1 Command

### 2.1.1 Basic Commands

| Command Table | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **D/C#** | **HEX** | **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** | **Command** | **Description** |
| 0 | 0x15 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | Set | A [15:0] : Start Address (reset = 0) |
| 1 | $A_H$ | $A_{15}$ | $A_{14}$ | $A_{13}$ | $A_{12}$ | $A_{11}$ | $A_{10}$ | $A_9$ | $A_8$ | Column Address | B [15:0] : End Address (reset = 0) |
| 1 | $A_L$ | $A_7$ | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ | | |
| 1 | $B_H$ | $B_{15}$ | $B_{14}$ | $B_{13}$ | $B_{12}$ | $B_{11}$ | $B_{10}$ | $B_9$ | $B_8$ | | Range A from 0 to Display X-Size -1 |
| 1 | $B_L$ | $B_7$ | $B_6$ | $B_5$ | $B_4$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ | | |
| 0 | 0x75 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | Set | A [15:0] : Start Address (reset = 0) |
| 1 | $A_H$ | $A_{15}$ | $A_{14}$ | $A_{13}$ | $A_{12}$ | $A_{11}$ | $A_{10}$ | $A_9$ | $A_8$ | Row Address | B [15:0] : End Address (reset = 0) |
| 1 | $A_L$ | $A_7$ | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ | | |
| 1 | $B_H$ | $B_{15}$ | $B_{14}$ | $B_{13}$ | $B_{12}$ | $B_{11}$ | $B_{10}$ | $B_9$ | $B_8$ | | Range A from 0 to Display Y-Size -1 |
| 1 | $B_L$ | $B_7$ | $B_6$ | $B_5$ | $B_4$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ | | |
| 0 | 0x5C | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | Write<br>RAM Command | Enable MCU to write Data into VRAM |
| 0 | 0x5D | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | Read<br>RAM Command | Enable MCU to read Data from VRAM |
| 0 | 0xA0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | Set | A[0] = 0b, Horizontal address increment (reset) |
| 1 | A [7:0] | $A_7$ | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ | Display Setup | A[0] = 1b, Vertical address increment<br><br>A[1] = 0b, Column Scan Left to Right [ L → R ] (reset)<br>A[1] = 1b, Column Scan Right to Left [ L ← R ]<br><br>A[2] = 0b, Row Scan Up to Down [ U → D ] (reset)<br>A[2] = 1b, Row Scan Down to UP [ U ← D ]<br><br>A[3] = 0b, reserved<br>A[3] = 1b, reserved<br><br>A[4] = 0b, reserved<br>A[4] = 1b, reserved<br><br>A[5] = 0b, Read/Write Display Data from/to Page 1<br>A[5] = 1b, Read/Write Display Data from/to Page 2<br><br>A[6] = 0b, Display Data Page 1<br>A[6] = 1b, Display Data Page 2<br><br>A[7] = 0b, reserved<br>A[7] = 1b, reserved |
| 0 | 0xAB | | | | | | | | | Set<br>Display Function | |
| 0 | 0x18 | | | | | | | | | Set<br>Read Address | |
| | | | | | | | | | | | |

### 2.2.1 Set Column Address (0x15)

This five byte command specifies column start address and end address of the display data RAM. This command also sets the column address pointer to column start address. This pointer is used to define the current read/write column address in graphic display data RAM. If horizontal address increment mode is enabled by command A0h, after finishing read/write one column data, it is incremented automatically to the next column address. Whenever the column address pointer finishes accessing the end column address, it is reset back to start column address and the row address is incremented to the next row.

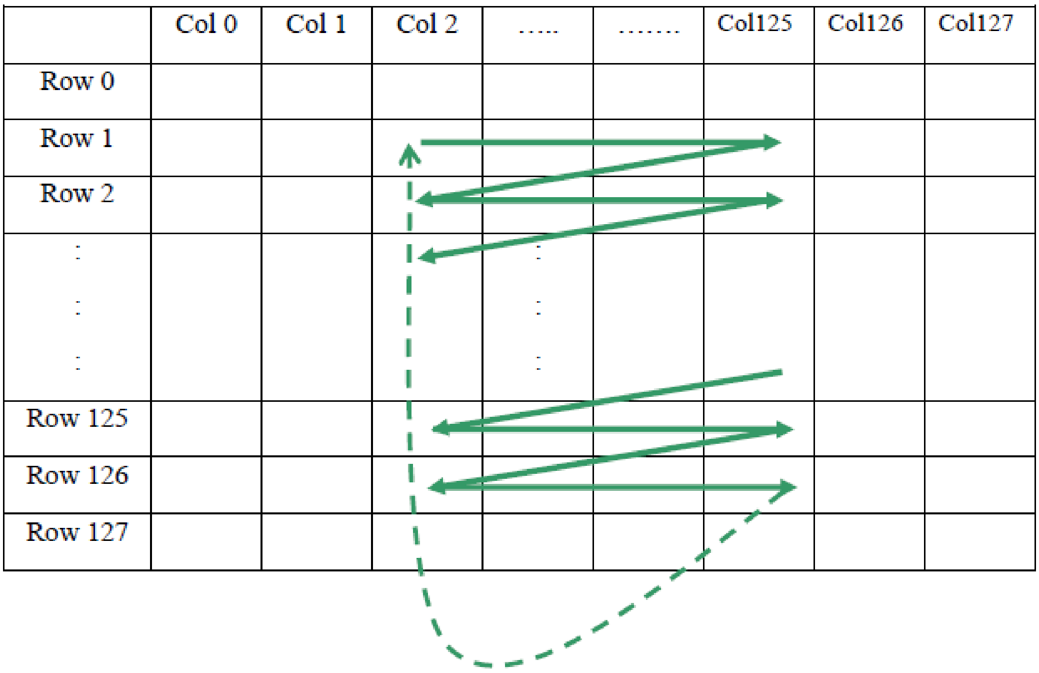
### 2.2.2 Set Row Address (0x75)

This five byte command specifies row start address and end address of the display data RAM. This command also sets the row address pointer to row start address. This pointer is used to define the current read/write row address in graphic display data RAM. If vertical address increment mode is enabled by command A0h, after finishing read/write one row data, it is incremented automatically to the next row address. Whenever the row address pointer finishes accessing the end row address, it is reset back to start row address.

For example, column start address is set to 2 and column end address is set to 125, row start address is set to 1 and row end address is set to 126. Horizontal address increment mode is enabled by command A0h. In this case, the graphic display data RAM column accessible range is from column 2 to column 125 and from row 1 to row 126 only. In addition, the column address pointer is set to 2 and row address pointer is set to 1.

After finishing read/write one pixel of data, the column address is increased automatically by 1 to access the next RAM location for next read/write operation (*solid line in Figure 2-1*).
Whenever the column address pointer finishes accessing the end column 125, it is reset back to column 2 and row address is automatically increased by 1 (*solid line in Figure 2-1*).
While the end row 126 and end column 125 RAM location is accessed, the row address is reset back to 1 and the column address is reset back to 2 (*dotted line in Figure 2-1*).

**Figure 2-1 : Example of Column and Row Address Pointer Movement**



|  | Col 0 | Col 1 | Col 2 | ….. | ……. | Col125 | Col126 | Col127 |
|---|---|---|---|---|---|---|---|---|
| Row 0 |  |  |  |  |  |  |  |  |
| Row 1 |  |  |  |  |  |  |  |  |
| Row 2 |  |  |  |  |  |  |  |  |
| :<br>:<br>: |  |  |  | :<br>:<br>: |  |  |  |  |
| Row 125 |  |  |  |  |  |  |  |  |
| Row 126 |  |  |  |  |  |  |  |  |
| Row 127 |  |  |  |  |  |  |  |  |

### 2.2.3 Write RAM Command (0x5C)

After entering this single byte command, data entries will be written into the display RAM until the CS# line is entered High pegel again or another command is written. Address pointer is increased accordingly. This command must be sent before write data into RAM.

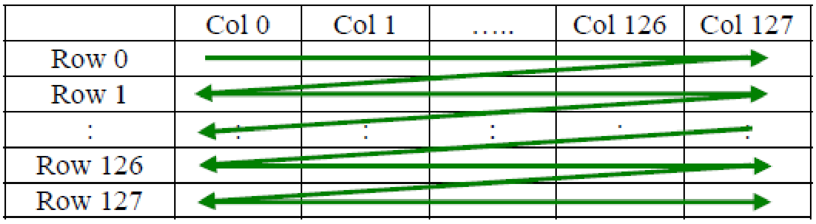
### 2.2.4 Read RAM Command (0x5C)

After entering this single byte command, poll the wait line to ensure that the data to reading becomes valid. If the Wait line is Low you can read data from display RAM until the Wait line, CS# line is entered High pegel again or another command is written. Address pointer is increased accordingly. This command must be sent before read data from RAM.

### 2.2.5 Set Display Setup

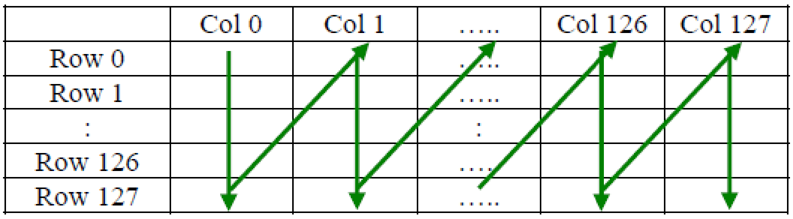
- Address increment mode (A[0])
  When A[0] is set to 0, the driver is set as horizontal address increment mode. After the display RAM is read / written, the column address pointer is increased automatically by 1. If the column address pointer reaches column end address, the column address pointer is reset to column start address and row address pointer is increased by 1. The sequence of movement of the row and column address point for horizontal address increment mode is shown in *Figure 2-2*.

**Figure 2-2 : Address Pointer Movement of Horizontal Address Increment Mode**



When A[0] is set to 1, the driver is set to vertical address increment mode. After the display RAM is read / written, the row address pointer is increased automatically by 1. If the row address pointer reaches the row end address, the row address pointer is reset to row start address and column address pointer is increased by 1. The sequence of movement of the row and column address point for vertical address increment mode is shown in *Figure 2-3*.

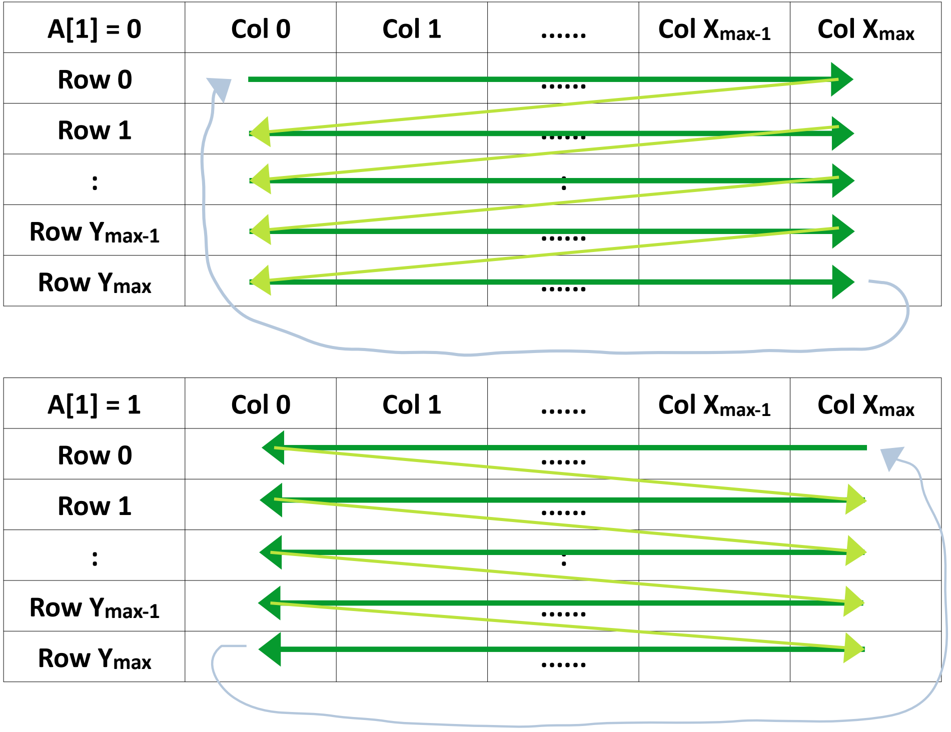
**Figure 2-3 : Address Pointer Movement of  Vertical Address Increment Mode**



- Column Scan (A[1])
  This command bit is made for increasing the layout flexibility of segment scan signals in Display module with segment arranged from left to right (when A[1] is set to 0) or vice versa (when A[1] is set to 1), as demonstrated in *Figure 2-4.*

A[1] = 0: RAM Column 0 → Colomn $X_{max}$ maps to Display Column 0 → Column $X_{max}$
A[1] = 1: RAM Column 0 → Colomn $X_{max}$ maps to Display Column $X_{max}$ → Column 0

**Figure 2-4 : Column Scan - Ram to Display**

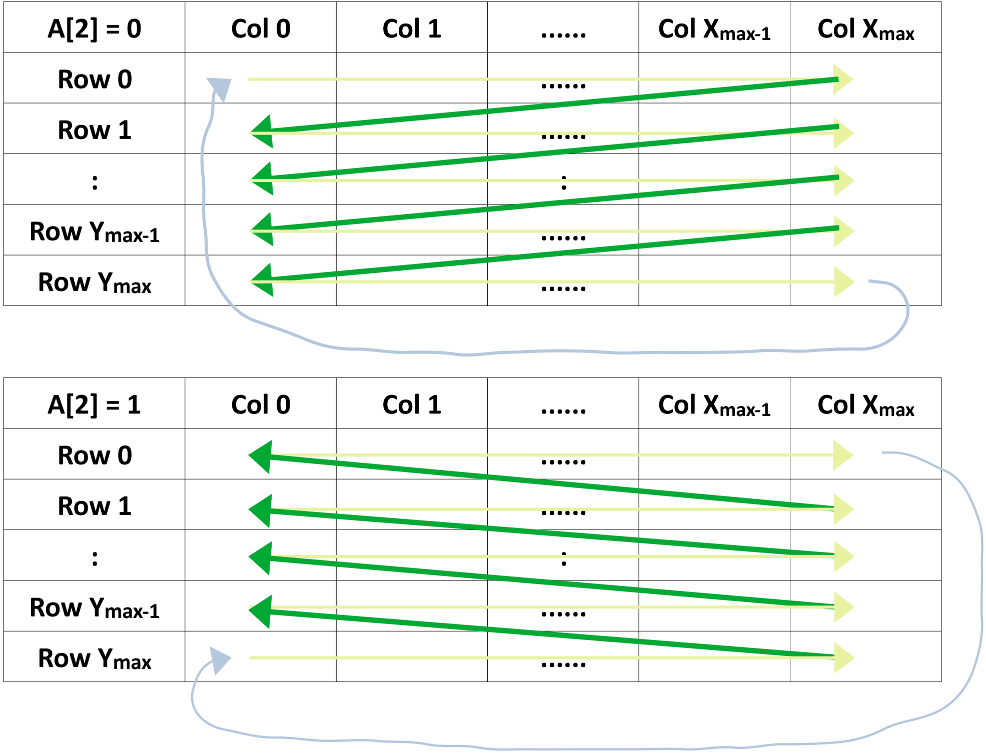


| A[1] = 0 | Col 0 | Col 1 | ...... | Col $X_{max-1}$ | Col $X_{max}$ |
|---|---|---|---|---|---|
| Row 0 | | | ...... | | |
| Row 1 | | | ...... | | |
| : | | | : | | |
| Row $Y_{max-1}$ | | | ...... | | |
| Row $Y_{max}$ | | | ...... | | |

| A[1] = 1 | Col 0 | Col 1 | ...... | Col $X_{max-1}$ | Col $X_{max}$ |
|---|---|---|---|---|---|
| Row 0 | | | ...... | | |
| Row 1 | | | ...... | | |
| : | | | : | | |
| Row $Y_{max-1}$ | | | ...... | | |
| Row $Y_{max}$ | | | ...... | | |

- Row Scan (A[2])
  This command bit is made for increasing the layout flexibility of segment scan signals in Display module with rows arranged from up to down (when A[2] is set to 0) or vice versa (when A[2] is set to 1), as demonstrated in *Figure 2-5.*
  A[2] = 0: RAM Row 0 → Row $Y_{max}$ maps to Display Row 0 → Row $Y_{max}$
  A[2] = 1: RAM Row 0 → Row $Y_{max}$ maps to Display Row $Y_{max}$ → Row 0

**Figure 2-5 : Row Scan - Ram to Display**



| A[2] = 0 | Col 0 | Col 1 | ...... | Col $X_{max-1}$ | Col $X_{max}$ |
|---|---|---|---|---|---|
| Row 0 | | | ...... | | |
| Row 1 | | | ...... | | |
| : | | | : | | |
| Row $Y_{max-1}$ | | | ...... | | |
| Row $Y_{max}$ | | | ...... | | |

| A[2] = 1 | Col 0 | Col 1 | ...... | Col $X_{max-1}$ | Col $X_{max}$ |
|---|---|---|---|---|---|
| Row 0 | | | ...... | | |
| Row 1 | | | ...... | | |
| : | | | : | | |
| Row $Y_{max-1}$ | | | ...... | | |
| Row $Y_{max}$ | | | ...... | | |