Application Note – LKEB-G02-A ↔ LKG-128064-D2 (AN01)

(1) Program – 128064D2

(2) Source Code – Assembly Language

```
;*************************************************************************
;*                                                                       *
;* Program     : 128064-D2                                               *
;* Title       : 128 column x 64 row LCD Graphic Module Driving Program  *
;*                                                                       *
;* MCU         : AT89C51, 12MHz                                          *
;* Programmer  : Stephen Yu, Aug. 2002                                   *
;* File name   : 128064D2.asm                                            *
;* Eval. Board : LKEB-G02-A                                              *
;* Description : Drive 128x64 LCD graphic module with LCD driver         *
;*               KS0108 x2 + KS0107 x1                                   *
;* LCD Module  : LKG-128064-D2 or equivalent                             *
;* Assembler   : AD2500 or equivalent                                    *
;* Copyright   : Lcdmod Kit                                              *
;*************************************************************************

;*************************************
;*      MCU <-> LCM pin connection     *
;*                                     *
;* P1.0 - 1.7 <-> DB0 -DB7             *
;*************************************

        ORG     0H

CSA     REG     P3.4            ; PIN REGISTRATION
CSB     REG     P3.5
RS      REG     P3.3
RW      REG     P3.1
E       REG     P3.0
RST     REG     P3.6
EL1     REG     P3.7
CLICK   REG     P3.2


COM     EQU     20H
DAT     EQU     21H

START:                          ; MAIN PROGRAM
        MOV     P1,#00H         ; CLEAR PORT 1 AND 3
        MOV     P3,#00H
        NOP
        NOP
        SETB    RST             ; SET THE RST PIN TO HIGH
        SETB    CLICK           ; SET THE 8051 P3.2 TO HIGH

        CLR     EL1             ; TURN ON THE EL BACKLIGHT

        CALL    FULL_OFF        ; TURN DISPLAY OFF
        CALL    TURN_ON         ; TURN DISPLAY ON

        CALL    FULL_OFF        ; ALL PIXEL OFF
        CALL    BUTTON

        CALL    FULL_ON         ; ALL PIXEL ON
        CALL    BUTTON
```

```
        CALL    ROW
        CALL    BUTTON          ; ALTERNATE ROW ON

        CALL    COLUMN          ; ALTERNATE COLUMN ON
        CALL    BUTTON

        CALL    S_CHECKER       ; 1x1 CHECKER BOARD
        CALL    BUTTON

        CALL    M_CHECKER       ; 2x2 CHECKER BOARD
        CALL    BUTTON

        CALL    B_CHECKER       ; 4x4 CHECKER BOARD
        CALL    BUTTON

        CALL    GRAPH_R         ; DISPLAY AN IMAGE
        CALL    GRAPH_L
        CALL    BUTTON
        JMP     START

TURN_ON:
        MOV     COM,#0C0H       ; DISPLAY START LINE = 0
        LCALL   INST_L
        LCALL   INST_R
        MOV     COM,#03FH       ; DISPLAY ON
        LCALL   INST_L
        LCALL   INST_R
        RET

TURN_OFF:
        MOV     COM,#0C0H       ; DISPLAY START LINE = 0
        LCALL   INST_L
        LCALL   INST_R
        MOV     COM,#03EH       ; DISPLAY OFF
        LCALL   INST_L
        LCALL   INST_R
        RET


FULL_OFF:                       ; ( PROCEDURE FOR ALL PIXEL OFF )
        MOV     R4,#00H
FULL_OFF1:
        MOV     A,R4
        ORL     A,#0B8H         ; SET PAGE ADDRESS TO PAGE 0
        MOV     COM,A
        LCALL   INST_L
        LCALL   INST_R
        MOV     COM,#40H        ; SET Y ADDRESS TO 1
        LCALL   INST_L
        LCALL   INST_R
        MOV     R3,#40H         ; SET Y ADDRESS LOOP COUNTER = 64
FULL_OFF2:
        MOV     DAT,#00H        ; PUT 00000000 TO THE DATA BYTE
        LCALL   DATA_L
        LCALL   DATA_R
        DJNZ    R3,FULL_OFF2
        INC     R4                      ; PAGE ADDRESS LOOP COUNTER (0-7)
        CJNE    R4,#08H,FULL_OFF1       ; COMPARE JUMP IF NOT EQUAL
        RET

FULL_ON:                        ; ( PROCEDURE FOR ALL PIXEL OFF )
        MOV     R4,#00H
FULL_ON1:
        MOV     A,R4
        ORL     A,#0B8H         ; SET PAGE ADDRESS TO PAGE 0
        MOV     COM,A
        LCALL   INST_L
```



```
        LCALL   INST_R
        MOV     COM,#040H     ; SET Y ADDRESS TO 1
        LCALL   INST_L
        LCALL   INST_R
        MOV     R3,#040H      ; SET Y ADDRESS LOOP COUNTER = 64
FULL_ON2:
        MOV     DAT,#0FFH     ; PUT 11111111 TO THE DATA BYTE
        LCALL   DATA_L
        LCALL   DATA_R
        DJNZ    R3,FULL_ON2
        INC     R4                  ; PAGE ADDRESS LOOP COUNTER (0-7)
        CJNE    R4,#08H,FULL_ON1    ; COMPARE JUMP IF NOT EQUAL
        RET

ROW:                          ; ( PROCEDURE FOR ALTERNATE ROW ON )
        MOV     R4,#00H
ROW1:
        MOV     A,R4
        ORL     A,#0B8H       ; SET PAGE ADDRESS TO PAGE 0
        MOV     COM,A
        LCALL   INST_L
        LCALL   INST_R
        MOV     COM,#040H     ; SET Y ADDRESS TO 1
        LCALL   INST_L
        LCALL   INST_R
        MOV     R3,#040H      ; SET Y ADDRESS LOOP COUNTER = 64
ROW2:
        MOV     DAT,#0AAH     ; PUT 10101010 TO THE DATA BYTE
        LCALL   DATA_L
        LCALL   DATA_R
        DJNZ    R3,ROW2
        INC     R4                ; PAGE ADDRESS LOOP COUNTER (0-7)
        CJNE    R4,#08H,ROW1      ; COMPARE JUMP IF NOT EQUAL
        RET

COLUMN:                       ; ( PROCEDURE FOR ALTERNATE COLUMN ON )
        MOV     R4,#00H
COLUMN1:
        MOV     A,R4
        ORL     A,#0B8H       ; SET PAGE ADDRESS TO PAGE 0
        MOV     COM,A
        LCALL   INST_L
        LCALL   INST_R
        MOV     COM,#040H     ; SET Y ADDRESS TO 1
        LCALL   INST_L
        LCALL   INST_R
        MOV     R3,#020H      ; SET Y ADDRESS LOOP COUNTER = 32
COLUMN2:
        MOV     DAT,#0FFH     ; PUT 11111111 TO THE DATA BYTE
        LCALL   DATA_L
        LCALL   DATA_R
        MOV     DAT,#00H      ; PUT 00000000 TO THE DATA BYTE
        LCALL   DATA_L
        LCALL   DATA_R
        DJNZ    R3,COLUMN2
        INC     R4                  ; PAGE ADDRESS LOOP COUNTER (0-7)
        CJNE    R4,#08H,COLUMN1     ; COMPARE JUMP IF NOT EQUAL
        RET

S_CHECKER:                    ; ( PROCEDURE FOR 1x1 CHECKER BOARD )
        MOV     R4,#00H
S_CHECKER1:
        MOV     A,R4
        ORL     A,#0B8H       ; SET PAGE ADDRESS TO PAGE 0
        MOV     COM,A
        LCALL   INST_L
        LCALL   INST_R
        MOV     COM,#040H     ; SET Y ADDRESS TO 1
```



```
        LCALL   INST_L
        LCALL   INST_R
        MOV     R3,#020H        ; SET Y ADDRESS LOOP COUNTER = 32
S_CHECKER2:
        MOV     DAT,#0AAH       ; PUT 10101010 TO THE DATA BYTE
        LCALL   DATA_L
        LCALL   DATA_R
        MOV     DAT,#055H       ; PUT 01010101 TO THE DATA BYTE
        LCALL   DATA_L
        LCALL   DATA_R
        DJNZ    R3,S_CHECKER2
        INC     R4                      ; PAGE ADDRESS LOOP COUNTER (0-7)
        CJNE    R4,#08H,S_CHECKER1  ; COMPARE JUMP IF NOT EQUAL
        RET

M_CHECKER:                      ; ( PROCEDURE FOR 2x2 CHEKCER BOARD )
        MOV     R4,#00H
M_CHECKER1:
        MOV     A,R4
        ORL     A,#0B8H         ; SET PAGE ADDRESS TO PAGE 0
        MOV     COM,A
        LCALL   INST_L
        LCALL   INST_R
        MOV     COM,#040H       ; SET Y ADDRESS TO 1
        LCALL   INST_L
        LCALL   INST_R
        MOV     R3,#010H        ; SET Y ADDRESS LOOP COUNTER = 16
M_CHECKER2:
        MOV     DAT,#0CCH       ; PUT 11001100 TO THE DATA BYTE
        LCALL   DATA_L
        LCALL   DATA_R
        LCALL   DATA_L
        LCALL   DATA_R
        MOV     DAT,#033H       ; PUT 00110011 TO THE DATA BYTE
        LCALL   DATA_L
        LCALL   DATA_R
        LCALL   DATA_L
        LCALL   DATA_R
        DJNZ    R3,M_CHECKER2
        INC     R4                      ; PAGE ADDRESS LOOP COUNTER (0-7)
        CJNE    R4,#08H,M_CHECKER1  ; COMPARE JUMP IF NOT EQUAL
        RET

B_CHECKER:                      ; ( PROCEDURE FOR 4x4 CHECKER BOARD )
        MOV     R4,#00H
B_CHECKER1:
        MOV     A,R4
        ORL     A,#0B8H         ; SET PAGE ADDRESS TO PAGE 0
        MOV     COM,A
        LCALL   INST_L
        LCALL   INST_R
        MOV     COM,#040H       ; SET Y ADDRESS TO 1
        LCALL   INST_L
        LCALL   INST_R
        MOV     R3,#08H         ; SET Y ADDRESS LOOP COUNTER = 8
B_CHECKER2:
        MOV     DAT,#0F0H       ; PUT 11110000 TO THE DATA BYTE
        LCALL   DATA_L
        LCALL   DATA_R
        LCALL   DATA_L
        LCALL   DATA_R
        LCALL   DATA_L
        LCALL   DATA_R
        LCALL   DATA_L
        LCALL   DATA_R
        MOV     DAT,#0FH        ; PUT 00001111 TO THE DATA BYTE
        LCALL   DATA_L
        LCALL   DATA_R
```



```
        LCALL   DATA_L
        LCALL   DATA_R
        LCALL   DATA_L
        LCALL   DATA_R
        LCALL   DATA_L
        LCALL   DATA_R
        DJNZ    R3,B_CHECKER2
        INC     R4                  ; PAGE ADDRESS LOOP COUNTER (0-7)
        CJNE    R4,#08H,B_CHECKER1  ; COMPARE JUMP IF NOT EQUAL
        RET

GRAPH_L:                            ; ( PORCEDURE FOR LEFT SCREEN IMAGE )
        MOV     DPTR,#IMG_0L
        MOV     R4,#00H
GRAPH_L1:
        MOV     A,R4
        ORL     A,#0B8H             ; SET PAGE ADDRESS TO PAGE 0
        MOV     COM,A
        LCALL   INST_L
        MOV     COM,#040H           ; SET Y ADDRESS TO 1
        LCALL   INST_L
        MOV     R3,#040H            ; SET Y ADDRESS LOOP COUNTER = 64
GRAPH_L2:
        MOV     A,#00H
        MOVC    A,@A + DPTR         ; GET THE IMAGE DATA
        MOV     DAT,A
        LCALL   DATA_L
        INC     DPTR
        DJNZ    R3,GRAPH_L2
        INC     R4
        CJNE    R4,#08H,GRAPH_L1
        RET

GRAPH_R:                            ; ( PROCEDURE FOR RIGHT SCREEN IMAGE )
        MOV     DPTR,#IMG_0R
        MOV     R4,#00H
GRAPH_R1:
        MOV     A,R4
        ORL     A,#0B8H             ; SET PAGE ADDRESS TO PAGE 0
        MOV     COM,A
        LCALL   INST_R
        MOV     COM,#040H           ; SET Y ADDRESS TO 1
        LCALL   INST_R
        MOV     R3,#040H            ; SET Y ADDRESS LOOP COUNTER = 64
GRAPH_R2:
        MOV     A,#00H
        MOVC    A,@A + DPTR         ; GET THE IMAGE DATA
        MOV     DAT,A
        LCALL   DATA_R
        INC     DPTR
        DJNZ    R3,GRAPH_R2
        INC     R4
        CJNE    R4,#08H,GRAPH_R1
        RET

INST_L: CLR     CSA                 ; ( PROCEDURE FOR LEFT SCREEN INSTRUCTION WRITE )
        SETB    CSB
        CLR     RS
        SETB    RW
INST_L1:
        MOV     P1,#0FFH
        NOP
        SETB    E
        NOP
        MOV     A,P1
        NOP
        CLR     E
        NOP
```



```
        JB      ACC.7,INST_L1   ; CHECK BUSY
        CLR     RW
        MOV     P1,COM
        NOP
        SETB    E
        NOP
        CLR     E
        NOP
        RET

DATA_L: CLR     CSA             ; ( PROCEDURE FOR LEFT SCREEN DATA WRITE )
        SETB    CSB
        CLR     RS
        SETB    RW
DATA_L1:
        MOV     P1,#0FFH
        NOP
        SETB    E
        NOP
        MOV     A,P1
        NOP
        CLR     E
        NOP
        JB      ACC.7,DATA_L1   ; CHECK BUSY
        SETB    RS
        CLR     RW
        MOV     P1,DAT
        NOP
        SETB    E
        NOP
        CLR     E
        NOP
        RET

INST_R: SETB    CSA             ; (PROCEDURE FOR RIGHT SCREEN INSTRUCTION WRITE )
        CLR     CSB
        CLR     RS
        SETB    RW
INST_R1:
        MOV     P1,#0FFH
        NOP
        SETB    E
        NOP
        MOV     A,P1
        NOP
        CLR     E
        NOP
        JB      ACC.7,INST_R1   ; CHECK BUSY
        CLR     RW
        MOV     P1,COM
        NOP
        SETB    E
        NOP
        CLR     E
        NOP
        RET

DATA_R: SETB    CSA             ; ( PROCEDURE FOR RIGHT SCREEN DATA WRITE )
        CLR     CSB
        CLR     RS
        SETB    RW
DATA_R1:
        MOV     P1,#0FFH
        NOP
        SETB    E
        NOP
        MOV     A,P1
        NOP
```



```
        CLR     E
        NOP
        JB      ACC.7,DATA_R1   ; CHECK BUSY
        SETB    RS
        NOP
        CLR     RW
        MOV     P1,DAT
        NOP
        SETB    E
        NOP
        CLR     E
        NOP
        RET


BUTTON: JB      CLICK,$         ; ( PROCEDURE FOR DETECT CLICKING OF PUSH BUTTON )
        ACALL   WAIT            ; ESCAPE FROM THE DEBOUNCING OF BUTTON
        JNB     CLICK,$
        ACALL   WAIT
        RET

DELAY:                          ; TIME DEALY ROUTINE
        PUSH    6
        PUSH    5
        PUSH    4
        MOV     R6,#100
$2      MOV     R5,#100
$1      MOV     R4,#30
        DJNZ    R4,$
        DJNZ    R5,$1
        DJNZ    R6,$2
        POP     4
        POP     5
        POP     6
        RET

WAIT:                           ; ( PROCEDURE FOR TIME WAITING )
        PUSH    6
        PUSH    7
        MOV     R6,#100
WAIT2:  MOV     R7,#200
        DJNZ    R7,$
        DJNZ    R6,WAIT2
        POP     7
        POP     6
        RET

IMG_0R  DB 00H,00H,00H,00H,00H,00H,A0H,20H;
        DB E0H,E0H,E0H,20H,A0H,80H,E0H,00H;
        DB 00H,00H,00H,00H,80H,80H,80H,80H;
        DB 80H,80H,80H,80H,00H,00H,80H,80H;
        DB 80H,A0H,20H,E0H,E0H,E0H,00H,E0H;
        DB 00H,00H,80H,80H,80H,80H,80H,80H;
        DB 80H,80H,80H,80H,80H,80H,80H,80H;
        DB 80H,00H,00H,00H,00H,00H,00H,80H;
        DB 00H,00H,00H,00H,00H,00H,A0H,A0H;
        DB BFH,BFH,BFH,A0H,AFH,A0H,BEH,BEH;
        DB BEH,80H,FEH,5FH,BFH,A1H,AEH,A2H;
        DB A1H,ABH,ABH,48H,3FH,1EH,7FH,BFH;
        DB B1H,F5H,B1H,BFH,BFH,BFH,A0H,AFH;
        DB 80H,E0H,A2H,A0H,BFH,BFH,BFH,A1H;
        DB 8DH,FFH,BFH,BFH,A1H,8DH,FFH,BFH;
        DB BFH,A1H,AEH,80H,E0H,0EH,5FH,BFH;
        DB 00H,00H,00H,00H,F0H,80H,40H,C0H;
        DB 40H,F0H,40H,40H,40H,F0H,40H,40H;
        DB 00H,C0H,80H,40H,40H,C0H,00H,40H;
        DB 00H,00H,C0H,30H,00H,C0H,30H,00H;
        DB 00H,C0H,00H,C0H,00H,00H,C0H,00H;
```

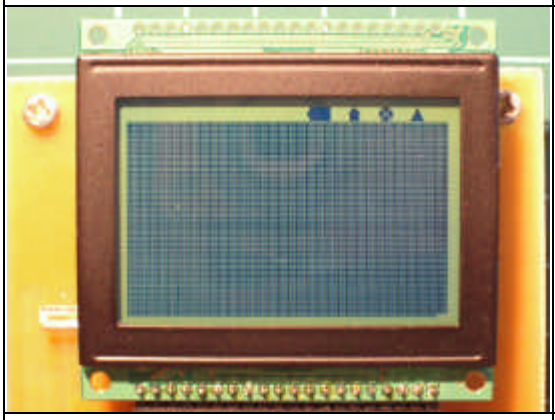
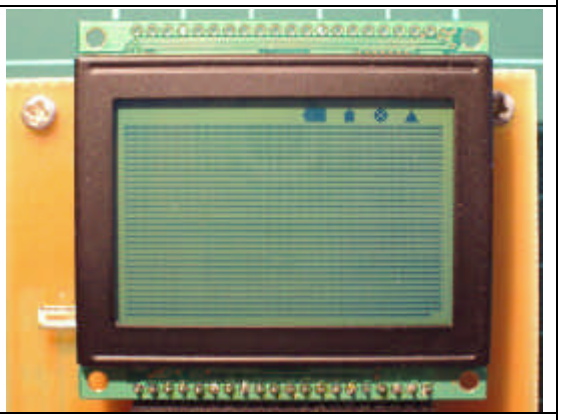
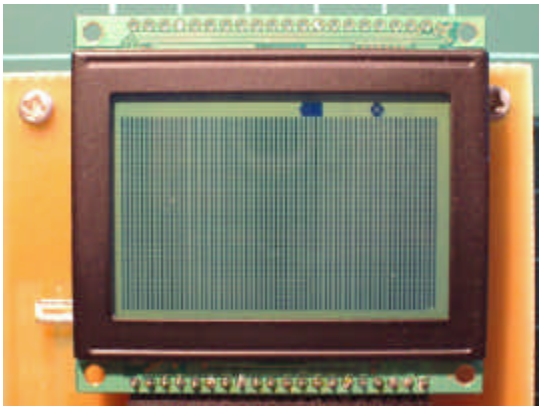
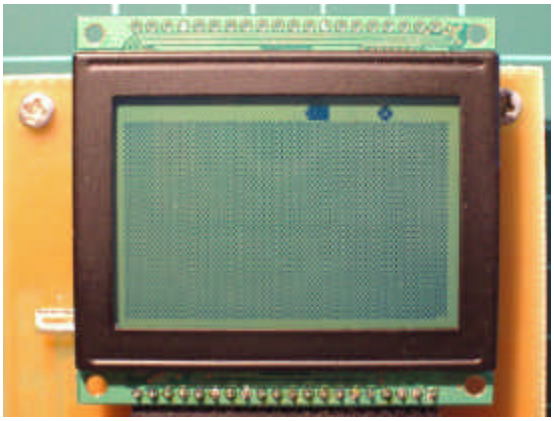
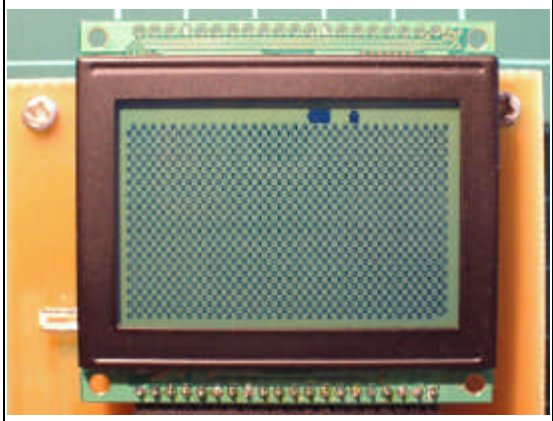
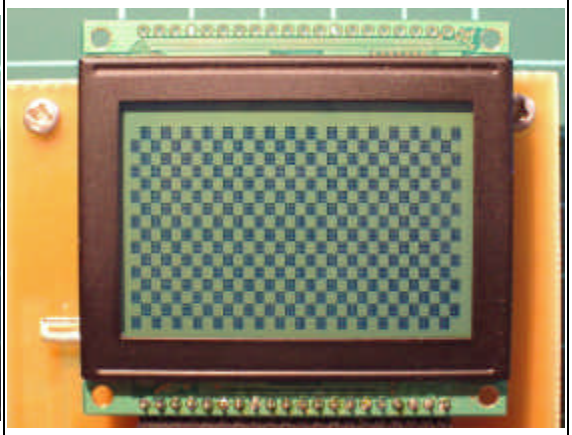


```
        DB C0H,00H,C0H,00H,00H,C0H,00H,C0H;
        DB 00H,C0H,00H,00H,C0H,00H,00H,00H;
        DB 00H,00H,F0H,00H,80H,40H,40H,C0H;
        DB 00H,00H,00H,00H,07H,00H,00H,07H;
        DB 00H,07H,04H,02H,00H,07H,04H,02H;
        DB 00H,3FH,04H,02H,02H,01H,00H,02H;
        DB 00H,0EH,01H,00H,0EH,01H,00H,00H;
        DB 00H,03H,07H,01H,06H,03H,00H,00H;
        DB 03H,07H,01H,06H,03H,00H,00H,03H;
        DB 07H,01H,06H,03H,00H,00H,04H,00H;
        DB 00H,00H,07H,03H,04H,04H,04H,02H;
        DB 00H,00H,00H,00H,00H,00H,00H,00H;
        DB 00H,00H,00H,00H,00H,00H,00H,00H;
        DB 00H,00H,00H,00H,00H,00H,00H,00H;
        DB 00H,00H,00H,00H,00H,00H,00H,00H;
        DB 00H,00H,00H,00H,00H,00H,00H,00H;
        DB E0H,F0H,38H,18H,08H,08H,18H,38H;
        DB F0H,E0H,00H,40H,C0H,C0H,40H,40H;
        DB C0H,C0H,00H,00H,00H,00H,00H,00H;
        DB 00H,00H,00H,00H,00H,00H,00H,00H;
        DB 00H,00H,00H,00H,00H,00H,00H,00H;
        DB 00H,00H,00H,00H,00H,00H,00H,00H;
        DB 00H,00H,00H,00H,00H,00H,00H,00H;
        DB 00H,00H,00H,00H,00H,00H,00H,00H;
        DB 03H,07H,0EH,0CH,08H,08H,0CH,0EH;
        DB 07H,03H,00H,08H,0FH,0FH,08H,08H;
        DB 0FH,0FH,08H,01H,01H,01H,01H,01H;
        DB 00H,00H,08H,08H,F8H,08H,08H,00H;
        DB 00H,E0H,10H,08H,08H,08H,08H,10H;
        DB 38H,00H,08H,F8H,08H,08H,08H,08H;
        DB 18H,30H,E0H,00H,00H,00H,00H,00H;
        DB 00H,40H,C0H,40H,40H,40H,C0H,40H;
        DB 40H,40H,80H,00H,80H,C0H,40H,40H;
        DB C0H,80H,00H,80H,C0H,40H,40H,C8H;
        DB F8H,00H,00H,40H,C0H,00H,00H,40H;
        DB 00H,00H,08H,08H,0FH,08H,08H,08H;
        DB 0FH,03H,06H,0CH,08H,08H,08H,0CH;
        DB 07H,02H,08H,0FH,08H,08H,08H,08H;
        DB 0CH,04H,03H,00H,00H,00H,00H,00H;
        DB 08H,08H,0FH,08H,08H,00H,0FH,08H;
        DB 00H,00H,0FH,08H,07H,0CH,08H,08H;
        DB 0CH,07H,00H,07H,0CH,08H,08H,0CH;
        DB 0FH,08H,00H,00H,0FH,08H,08H,08H;

IMG_0L  DB 80H,80H,80H,80H,80H,00H,00H,00H;
        DB 00H,80H,80H,80H,A0H,20H,E0H,E0H;
        DB E0H,00H,E0H,00H,00H,00H,00H,00H;
        DB 00H,00H,00H,A0H,20H,E0H,E0H,E0H;
        DB 20H,80H,E0H,80H,E0H,60H,A0H,A0H;
        DB A0H,C0H,80H,80H,A0H,A0H,A0H,80H;
        DB 60H,00H,80H,C0H,E0H,E0H,80H,A0H;
        DB 80H,00H,00H,00H,00H,00H,00H,00H;
        DB A0H,AEH,A0H,BFH,9FH,4FH,63H,1CH;
        DB 1EH,7FH,BFH,B1H,F5H,B1H,BFH,BFH;
        DB BFH,A0H,AFH,80H,E0H,00H,00H,00H;
        DB 00H,00H,00H,A0H,A0H,BFH,BFH,BFH;
        DB A2H,ABH,8BH,EBH,0EH,BFH,BDH,A8H;
        DB 80H,E0H,A2H,A0H,BFH,BFH,BFH,A0H;
        DB AFH,80H,E2H,7FH,BFH,BFH,A0H,AEH;
        DB 82H,00H,00H,00H,00H,00H,00H,00H;
        DB 00H,00H,80H,40H,40H,F0H,00H,00H;
        DB C0H,80H,C0H,80H,C0H,00H,00H,80H;
        DB C0H,40H,40H,80H,00H,00H,80H,40H;
        DB 40H,F0H,00H,00H,F0H,80H,80H,40H;
        DB 00H,C0H,40H,F0H,40H,40H,00H,00H;
        DB 00H,00H,00H,80H,40H,40H,C0H,00H;
        DB 80H,C0H,40H,40H,80H,00H,00H,C0H;
        DB 80H,C0H,80H,C0H,00H,00H,00H,00H;
```



```
DB 00H,07H,04H,04H,02H,07H,00H,00H;
DB 07H,00H,07H,00H,07H,00H,00H,03H;
DB 04H,04H,06H,03H,00H,07H,04H,04H;
DB 02H,07H,00H,00H,07H,01H,02H,04H;
DB 04H,07H,00H,07H,04H,02H,00H,04H;
DB 00H,00H,03H,04H,04H,04H,02H,00H;
DB 03H,04H,04H,06H,03H,00H,00H,07H;
DB 00H,07H,00H,07H,00H,00H,00H,00H;
DB 00H,08H,F8H,F8H,00H,00H,40H,C8H;
DB C8H,00H,00H,40H,C0H,C0H,40H,40H;
DB C0H,C0H,00H,80H,C0H,C0H,40H,40H;
DB C0H,80H,00H,00H,00H,00H,00H,00H;
DB 00H,00H,00H,00H,00H,00H,00H,00H;
DB 00H,00H,00H,00H,00H,00H,00H,00H;
DB 00H,00H,00H,00H,00H,00H,00H,00H;
DB 00H,00H,00H,00H,00H,00H,00H,00H;
DB 00H,08H,0FH,0FH,08H,00H,08H,0FH;
DB 0FH,08H,08H,08H,0FH,0FH,08H,08H;
DB 0FH,0FH,08H,07H,0FH,0DH,09H,09H;
DB 0DH,05H,00H,00H,00H,00H,00H,00H;
DB 00H,00H,00H,00H,00H,00H,00H,00H;
DB 00H,00H,00H,00H,00H,00H,00H,00H;
DB 00H,00H,00H,00H,00H,00H,00H,00H;
DB 00H,00H,00H,00H,00H,00H,00H,00H;
DB C0H,00H,08H,F8H,00H,00H,80H,C0H;
DB 40H,40H,40H,80H,00H,00H,00H,00H;
DB 08H,08H,F8H,88H,88H,88H,88H,70H;
DB 00H,80H,C0H,40H,40H,40H,80H,00H;
DB 40H,E0H,40H,00H,00H,40H,40H,40H;
DB 80H,00H,00H,40H,C8H,00H,00H,08H;
DB F8H,00H,00H,80H,C0H,40H,40H,40H;
DB 80H,00H,40H,C0H,80H,40H,40H,00H;
DB 0FH,08H,08H,0FH,08H,00H,07H,0DH;
DB 09H,09H,09H,05H,00H,00H,00H,00H;
DB 08H,08H,0FH,08H,08H,01H,06H,08H;
DB 08H,07H,0DH,09H,09H,09H,05H,00H;
DB 00H,0FH,08H,00H,06H,09H,09H,09H;
DB 0FH,08H,00H,08H,0FH,08H,08H,08H;
DB 0FH,08H,00H,07H,0DH,09H,09H,09H;
DB 05H,00H,08H,0FH,08H,08H,00H,00H;

END
```


(3) Picture Show

Full on	Row on
	
Column on	1 x 1 chekcer
	
2 x 2 checker	4 x 4 checker
	
Graph 1	Graph 2