

### Programming Sharp's Memory LCDs

Ken Green, Sharp Microelectronics of the Americas

#### INTRODUCTION

Sharp's Memory LCDs represent a step forward in power-saving monochrome displays. However, because the module's onboard controller is geared for power-saving functions, programming the module is done in a serial manner instead of the more familiar parallel manner in current use with color displays.

This Application Note will explain some general steps to programming the module, as well as give some practical examples.

#### VCOM

When designing with a memory LCD panel, a decision must be made as to how VCOM will be generated. VCOM is an alternating signal that prevents a DC bias from being built up within the panel. If this DC bias is allowed to accumulate, it will eventually reach the point where the state of the liquid crystals in the panel cells cannot be changed by the panel electronics and the cells will no longer change state.

Memory LCDs do not generate this signal internally. It must be supplied using one of two methods: software, or external clock. The mode is selected by the EXTMODE pin on the interface.

If external clock is selected [EXTMODE = H], the clock should be supplied on the EXTCOMM pin. See Figure 1. For further information on the requirements for amplitude and frequency for this clock signal, please see your particular part's Specifications sheet.

When the software clock is selected [EXTMODE = L], bit V of the command bit string sets the state of VCOM. See Figure 2. This bit must toggle (by writing to the panel) at least once per second.

Any command string can be used to change the state of the VCOM bit. If no update to the pixel memory is needed within a second, 'Change VCOM' can be used to toggle the bit state. In the descriptions and Figures, the V-bit represents the desired VCOM state when software VCOM control is selected. When the external clock is selected, then the V-bit state doesn't matter.

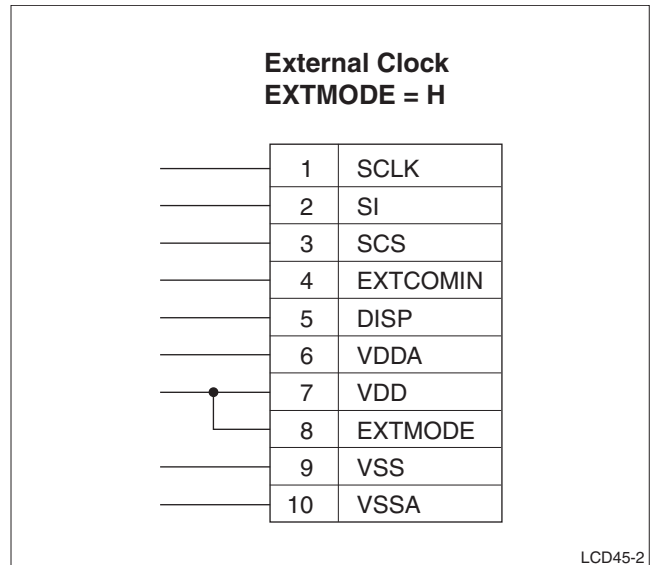


Figure 1. Selecting External Clock

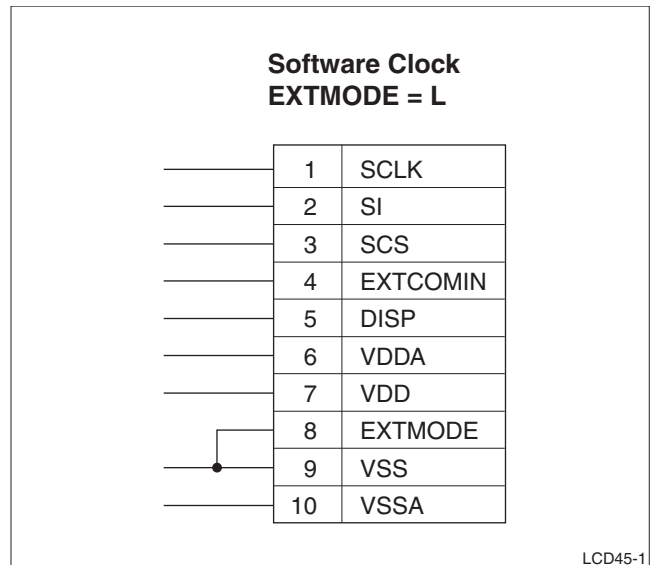


Figure 2. Selecting Internal Clock

## LCD COMMANDS

There are four commands that can be sent to a memory LCD panel:

- Write line
- Write multiple lines
- Change VCOM
- Clear screen

The commands are explained below. The illustrations are in little-endian format with the LSB on the left and the MSB on the right.

### Data Structure

Data is sent to the panel in little-endian format; with the LSB first. The data width for the Write Line and Write Multiple Line commands depends on the horizontal resolution of the panel itself. Therefore, if you're working with a panel having a resolution of  $400 \times 240$ , then the data width for this panel will require a minimum of 400 bits of data (plus overhead).

### Write Line

The minimum amount of data that can be written to the panel with the Write Line command is one line. The actual width of the data written depends on the horizontal resolution of the panel itself. Therefore, a panel with a resolution of  $400 \times 240$  will require a minimum of 400 bits of data (plus overhead). See Figure 3.

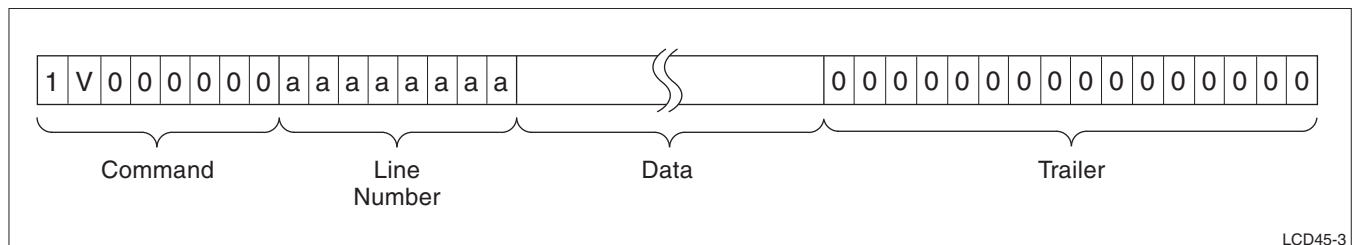
The command structure for Write Line is as follows:

- Command: 8 bits (including V-bit)
- Line address: 8 bits
- Data bits: leftmost pixel first, little-endian format, with data width depending on the resolution of the panel (as previously noted).
- Trailer: 16 bits. These clocks allow the panel time to transfer the data from the incoming latch to the pixel memory.

### Write Multiple Lines

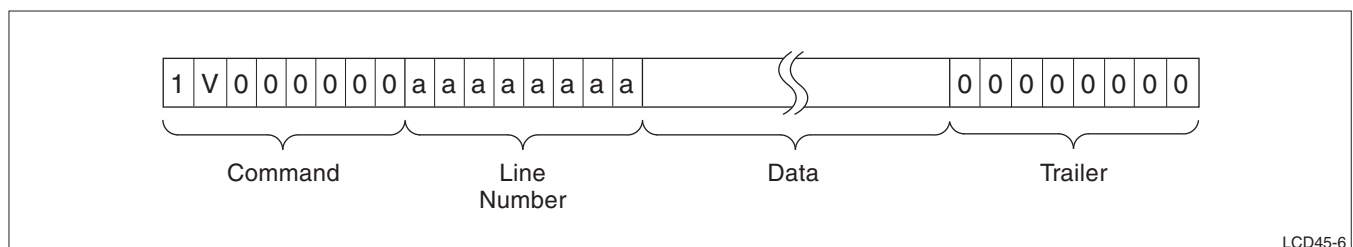
Multiple lines can be updated quickly with this command. A line address is still used so the lines do not have to be successive.

This command begins the same as the Write Line command; using the same Write Line command bits and data bits. See Figure 4.



LCD45-3

Figure 3. Write Line Data String



LCD45-6

Figure 4. Write Multiple Lines Data String, First Line

After the data bits follow 8 trailer bits (instead of 16), then the address of the next line (8 bits) to be written, followed by the data bits for that line, and so on until all of the desired lines are written. See Figure 5.

For the last line to be written, use 16 trailer bits (see Figure 6).

The command structure for Write Multiple Lines looks like this:

- Command: 8 bits (including V-bit)
- Line address: 8 bits
- Data bits: leftmost pixel first, little-endian format, with data width depending on the resolution of the panel (as explained previously).

- Trailer: 8 bits
- Address for next line
- Data for next line
- Trailer: 8 bits

This is carried on until all desired line updates except the last one have been sent. The structure for the last line is:

- Address for last line
- Data for last line
- Trailer: 16 bits

The 16-bit trailer allows for latch transfer time and signals the panel's controller that all line updates have been sent.

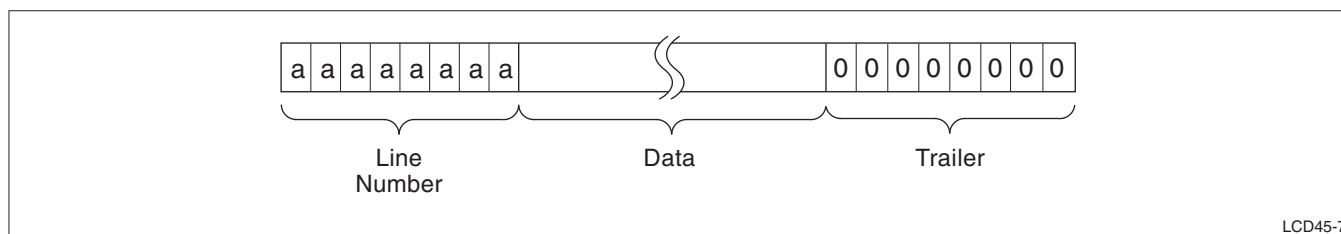


Figure 5. Write Multiple Lines Data String, Intermediate Lines

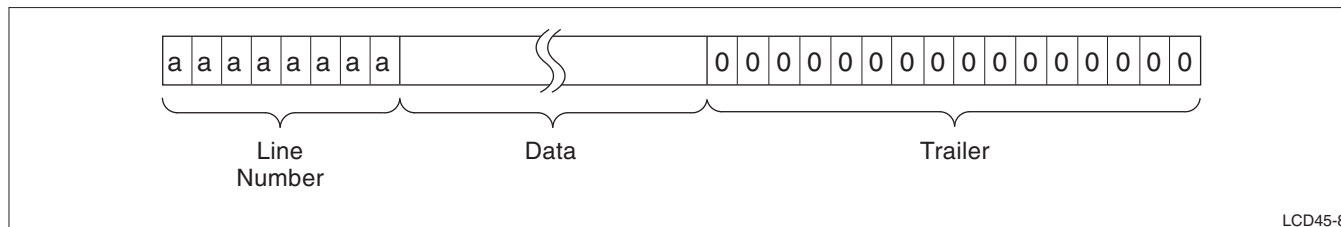


Figure 6. Write Multiple Lines Data String, Last Line

**Clear Screen**

This command clears the screen to all white by writing 0's to all of the memory locations in the frame buffer. See Figure 7. The command structure is as follows:

- Command: 8 bits (including V-bit)
- Trailer: 8 bits (allows for latch transfer time)

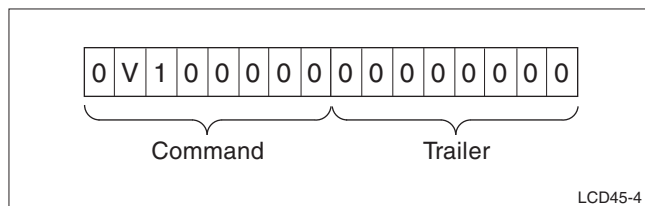


Figure 7. Clear Screen Data String

**Toggle VCOM**

This command is only used if [EXTMODE = L]. It is used to toggle the state of VCOM if no other command is to be executed. This is the periodic command to be sent within the appropriate amount of time to maintain proper VCOM frequency and DC panel bias. All commands contain the V-bit that allows toggling the state of VCOM. See Figure 8.

The command structure is:

- Command: 8 bits
- Trailer: 8 bits (allows for latch transfer time)

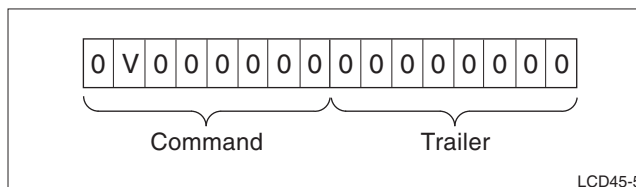


Figure 8. Toggle VCOM Data String

## SOFTWARE IMPLEMENTATION

This section contains code for interfacing with the panel. The SPI port should be set up so that when in the idle state, the serial chip select (SCS) and the serial clock (SC) are LOW. Data is clocked on the rising edge of SC. Care should be taken to meet all of the timing such as Setup and Hold times and SC clock speed as given in your part's Specifications.

It is important to note which bit (MSB or LSB) gets shifted out of the SPI port first. The code in the example here is written for an ARM Cortex M3 processor which shifts the MSB out first. Because the memory LCD expects the LSB first, the bits within each byte need to be swapped before being sent. This is done using the "swap" routine.

Also, because the particular processor picked has a DMA controller, it relieves the processor load of writing individual bytes. The Cortex DMA unit disables itself after the transfer is complete, then generates an interrupt using the SPI vector. The SPI interrupt vector should point to the routine "show\_frame". It is assumed that when show\_frame is called from an interrupt, that the processor context has already been saved; and that it will be restored when show\_frame exits.

The code can of course be modified if the processor picked doesn't have DMA or an SPI port.

```
// LCD commands - Note: the bits are reversed per the memory LCD data
// sheets because of the order the bits are shifted out in the SPI
// port.
#define MLCD_WR 0x80          //MLCD write line command
#define MLCD_CM 0x20          //MLCD clear memory command
#define MLCD_NO 0x00          //MLCD NOP command (used to switch VCOM)

//LCD resolution
#define MLCD_XRES 400        //pixels per horizontal line
#define MLCD_YRES 240        //pixels per vertical line
#define MLCD_BYTES_LINE MLCD_XRES / 8 //number of bytes in a line
#define MLCD_BUF_SIZE MLCD_YRES * MLCD_BYTES_LINE

//defines the VCOM bit in the command word that goes to the LCD
#define VCOM_HI 0x40
#define VCOM_LO 0x00

static char *frmbuffer; //current address of buffer to be displayed
static char locbuf[MLCD_BYTES_LINE + 3]; //local line buffer
static char linenum; //current line number being transmitted
static int stage = 0; //there are 3 stages in transmitting a buffer:
                    //stage 0: first line (has command in it)
                    //stage 1: 2nd through last line (no command)
                    //stage 2: null byte trailer
extern char vcom; //current state of vcom. This should alternate
                 //between VCOM_HI and VCOM_LO on a 1-30 second
                 //period.
```

```

////////////////////////////////////
//
// This routine transmits the contents of the pixel memory in
// a frame buffer to the memory LCD. It uses DMA to transfer
// each individual line. The address of the frame buffer to
// transfer is in the global variable show_frm.
//
// NOTE: this routine also acts as the interrupt handler for SPI
// interrupts.
//
// INPUTS:
//   The SPI and DMA units must have been previously initialized
//   show_frm:   pointer to the buffer to be displayed
//
////////////////////////////////////

void show_frame(char *show_frm) {
    int i;
    unsigned long sts;

    switch(stage) {
    case 0:    // stage 0: sending the first line. The command is
              // included in this line

                set_scs(HIGH);    //set SCS high

                frmbufptr = show_frm;    //init pointer to frame buffer
                linebuf = locbuf;    //init pointer to line buffer
                linenum = 1;    //init line address

                //first line of data is preceeded by a write command
                *linebuf++ = MLCD_WR | vcom;    //command (note: no need to swap)
                *linebuf++ = swap(linenum++);    //line number (address)

                for(i= 0; i < MLCD_BYTES_LINE; i++) *linebuf++ =
                    swap(*frmbufptr++);    //swap the order of the bits
                *linebuf++ = 0;    //trailer

                //Setup the SPI DMA controller (starting addr, size)
                TransferSetup(locbuf, linebuf - locbuf);

                //Start the xfer
                TransferStart;

                stage = 1;    //set to next stage
                break;
    }
}

```

```
case 1: //Sending a line (other than the first line). At this
        //point the DMA transfer for the previous line is done
        //and the channel disabled.

        linebuf = locbuf; //init pointer to line buffer
        *linebuf++ = swap(linenum++); //line number

        for(i= 0; i < MLCD_BYTES_LINE; i++) *linebuf++ =
            swap(*frmbufptr++); //swap the order of the bits
        *linebuf++ = 0; //trailer

        //set up DMA control
        TransferSetup(locbuf, linebuf - locbuf);

        //Start the xfer
        TransferStart;

        if(linenum > MLCD_YRES) stage = 2; //all lines sent, go to next
                                           //stage

        break;

case 2: //All lines sent, send a final null byte trailer. The DMA
        //transfer of the last line is finished and the channel
        //disabled. All that is left is to write a trailing null
        //byte. It's not worth using DMA to transfer 1 byte, so
        //it's done by directing writing to the SPI port.

        //Write the last (null) byte
        Write_to_SPI(0);

        //Since there is no interrupt on transmit buffer empty, loop
        //here until the byte is sent, then drop SCS.
        i = 0;
        while(SPI_BSY); //wait until done

        set_scs(LOW);
        stage = 0; //go back to stage 0 - sending out first line
    }
}
```

### POWER CONSIDERATIONS

One of the prime attributes of the memory LCD is low power operation. The actual power usage is directly related to how often data is written to the panel, and how often VCOM is toggled. We'll look at two scenarios and calculate power draw.

#### Power Scenario 1

Here all of the pixels will be written to the panel once per second. For the 1.35" panel, the quiescent current for the panel is 2.1  $\mu\text{A}$ . Each write to the panel (assuming that the entire panel is written), takes 360  $\mu\text{A}$ ; and lasts for about 5.38 ms (assuming a 2 MHz SPI clock speed). See Figure 9.

If averaged over a second, this amounts to an "average" current draw of approximately 2  $\mu\text{A}$ . Therefore, the panel would have an average power draw of 20.5  $\mu\text{W}$  (5 V operation). Note that a VCOM toggle can be done whenever the panel is written to including a data write.

#### Power Scenario 2

Here, we write all the pixels to the panel only every 30 seconds. The current to write to the panel "averaged" over that time is approximately 0.1  $\mu\text{A}$ . If VCOM is toggled once a second (using the Change VCOM command), the additional power draw for this action (assuming that it is done 29 times) is an average of 0.1  $\mu\text{A}$ . Therefore the average power draw over the 30-second period becomes 11.5  $\mu\text{W}$ . See Figure 10.

Applying these conditions to the 2.7" panel: writing all pixels to the whole panel once a second takes an average of 485  $\mu\text{W}$ . Writing all pixels once every 30 seconds (with a 1-second VCOM toggle) takes 135  $\mu\text{W}$ .

### SUMMARY

Sharp's Memory LCDs have a serial interface that makes them simple to program. The most challenging tasks for the programmer will be to ensure that VCOM is toggled periodically to maintain the lack of DC bias on the display; and that data is sent to the panel in the correct order.

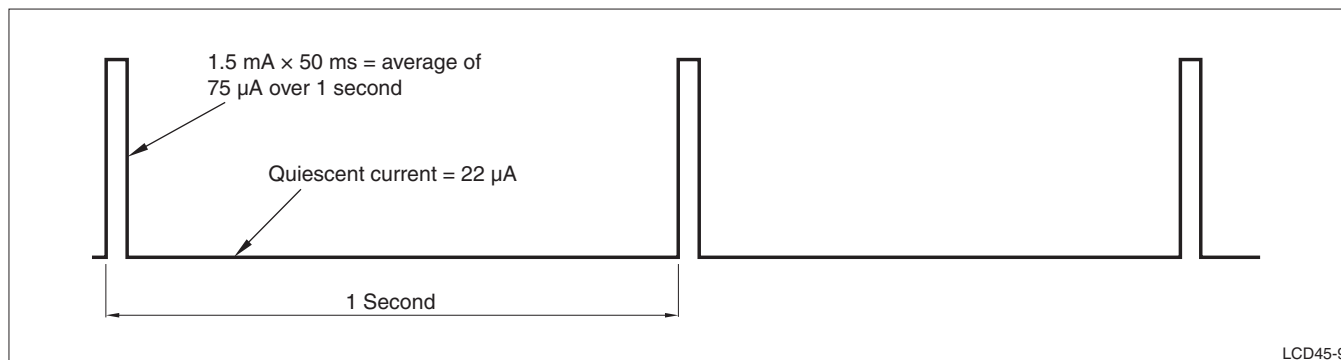


Figure 9. Writing All Pixels, Once per Second

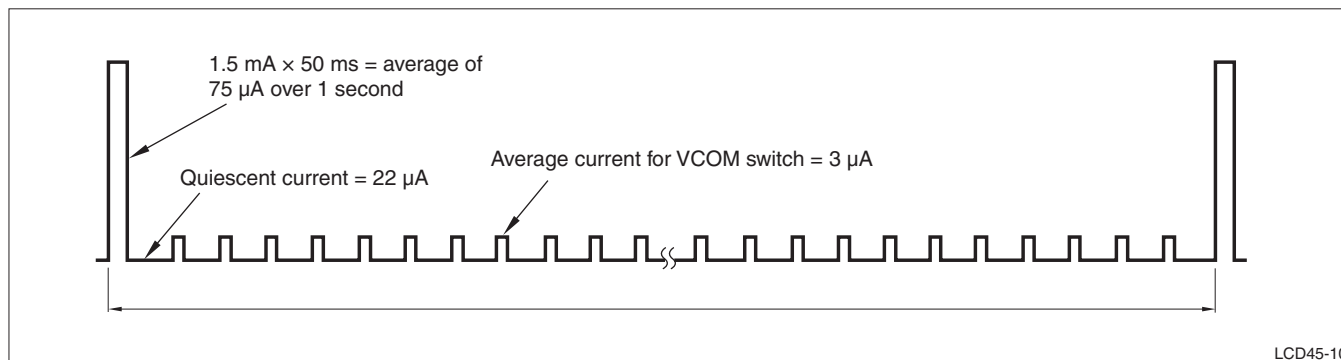


Figure 10. Writing All Pixels, Once Every 30 Seconds, Toggling VCOM Once Per Second

**SPECIFICATIONS ARE SUBJECT TO CHANGE WITHOUT NOTICE.**

Suggested applications (if any) are for standard use; See Important Restrictions for limitations on special applications. See Limited Warranty for SHARP's product warranty. The Limited Warranty is in lieu, and exclusive of, all other warranties, express or implied. ALL EXPRESS AND IMPLIED WARRANTIES, INCLUDING THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR USE AND FITNESS FOR A PARTICULAR PURPOSE, ARE SPECIFICALLY EXCLUDED. In no event will SHARP be liable, or in any way responsible, for any incidental or consequential economic or property damage.

**SHARP**<sup>®</sup>**NORTH AMERICA**

SHARP Microelectronics of the Americas  
5700 NW Pacific Rim Blvd.  
Camas, WA 98607, U.S.A.  
Phone: (1) 360-834-2500  
Fax: (1) 360-834-8903  
www.sharpsma.com

**EUROPE**

SHARP Microelectronics Europe  
Division of Sharp Electronics (Europe) GmbH  
Sonninstrasse 3  
20097 Hamburg, Germany  
Phone: (49) 40-2376-2286  
Fax: (49) 40-2376-2232  
www.sharpsme.com

**JAPAN**

SHARP Corporation  
Electronic Components & Devices  
22-22 Nagaike-cho, Abeno-Ku  
Osaka 545-8522, Japan  
Phone: (81) 6-6621-1221  
Fax: (81) 6117-725300/6117-725301  
www.sharp-world.com

**TAIWAN**

SHARP Electronic Components  
(Taiwan) Corporation  
8F-A, No. 16, Sec. 4, Nanking E. Rd.  
Taipei, Taiwan, Republic of China  
Phone: (886) 2-2577-7341  
Fax: (886) 2-2577-7326/2-2577-7328

**SINGAPORE**

SHARP Electronics (Singapore) PTE., Ltd.  
438A, Alexandra Road, #05-01/02  
Alexandra Technopark,  
Singapore 119967  
Phone: (65) 271-3566  
Fax: (65) 271-3855

**KOREA**

SHARP Electronic Components  
(Korea) Corporation  
RM 501 Geosung B/D, 541  
Dohwa-dong, Mapo-ku  
Seoul 121-701, Korea  
Phone: (82) 2-711-5813 ~ 8  
Fax: (82) 2-711-5819

**CHINA**

SHARP Microelectronics of China  
(Shanghai) Co., Ltd.  
28 Xin Jin Qiao Road King Tower 16F  
Pudong Shanghai, 201206 P.R. China  
Phone: (86) 21-5854-7710/21-5834-6056  
Fax: (86) 21-5854-4340/21-5834-6057

**Head Office:**

No. 360, Bashen Road,  
Xin Development Bldg. 22  
Waigaoqiao Free Trade Zone Shanghai  
200131 P.R. China  
Email: smc@china.global.sharp.co.jp

**HONG KONG**

SHARP-ROXY (Hong Kong) Ltd.  
3rd Business Division,  
17/F, Admiralty Centre, Tower 1  
18 Harcourt Road, Hong Kong  
Phone: (852) 28229311  
Fax: (852) 28660779  
www.sharp.com.hk

**Shenzhen Representative Office:**

Room 13B1, Tower C,  
Electronics Science & Technology Building  
Shen Nan Zhong Road  
Shenzhen, P.R. China  
Phone: (86) 755-3273731  
Fax: (86) 755-3273735