ADDENDUM

Kontron 7000CB Graphics Controller
Programmer's Manual

Revision 3.20 Software

The KON7K library diskette that accompanies your programmer's manual is the latest version: Software Revision 3.20, August 1988. The following files are on the revision 3.20 diskette:

7k.sys	System driver for the Kontron 7000CB;
kon7k.lib	Driver library for Microsoft C;
*.h	Include files (alias.h, init.h, kon7k.h, q.h, vram.h) that provide definitions and structures;
7k1*x*.dat	Hardware configuration data set for the K7000CB/1 board (1280 by 1024 by 8);
7k2*x*.dat	Hardware configuration data set for the K7000CB/2 board (1280 by 1024 by 4);
7k3*x*.dat	Hardware configuration data set for the K7000CB/3 board (1024 by 780 by 8);

7k4*x*.dat	Hardware configuration data set for the K7000CB/4 board (1024 by 780 by 4); *x* = s, m, or p for Sony, Mitsubishi, or Philips monitors, respectively;
7k.cfg	Executable form of configuration file for K7000CB;
setup7k.exe	Conversion program to generate 7k.cfg from the 7knx.dat file;
*.fnt	Bitmap text fonts;
egaon.exe;	Switches EGA driver on;
egaoff.exe	Switches EGA driver off;
egaset.exe	Configures and modifies the EGA driver parameter settings;
config.sys	Addendum to your standard config.sys, to allow loading the 7k.sys driver;
ctest.c, ctest.exe	Test image program;
tkon7k.exe	Toolbox program for generating various demo and benchmark graphic images;
default.lut	ASCII lookup table file.

Installing the KON7K Rev. 3.20 Software

1. If necessary, edit the 7k*nx*.dat software configuration data file to your specifications. Refer to Chapter 3 of the programmer's manual.

2. To install the software for your particular board version, import the 7k*nx*.dat file into the configuration file by executing the setup command

   ```
   setup7k 7knx.dat 7k.cfg
   ```

   where *n* represents the numbers 1 through 4, depending on the model of K7000CB contoller that you possess, and *x* represents the letters *s*, *m*, or *p* for Sony, Mitsubishi, or Philips monitors.

   For instance, if you want to install the Kontron 7000CB/4 board (1024 by 780 by 4 bit resolution), and you have a Philips monitor, enter

   ```
   setup7k 7k4p.dat 7k.cfg
   ```

3. Copy the driver, configuration, and text font files into the boot directory (typically C:\):

   7k.sys
   7k*.dat
   7k.cfg
   *.fnt

These files are sufficient for applications based on the standard KON7K library. However, specialized drivers, e.g., Computer Vision or AutoCAD, require their own configuration files:

7kadi.sys	(AutoCAD) or
7kcv.sys	(Computer Vision)

4. Also copy the configuration file and text font files into your current working directory:

   7k.cfg   *.fnt

Enhancements

The revision 3.20 software has the following enhancements not described in the programmer's manual:

- The k7_dpl, k7_pgfill, k7_xfm, and k7_xfmc functions were corrected.

- The dsp_zoom, dsp_pan, dsp_pick, and dsp_call display list functions now return an integer parameter, as follows:

1	Display list is valid;
0	Display list was erased because of overflow.

- There are nine new functions, described in the following pages.

dsp_switch

Switches display list output.

void	dsp_switch(*switch, path*);
int	*switch;*
char	*path;*

Allows you to choose an XRAM memory board extension of the graphics board, or use file access for display list maintenance (which is slower). This function switches display list output as follows:

switch = 0	XRAM of K7000CB
switch = 1	hard disk, file *path*

k7_getpt

Accesses parameters of the 7k.sys driver.

void	k7_getpt(*pts*);
struct P	*pts;*

Returns an array of pointers to the driver 7k.sys, defining all local data structures, which are also defined in vram.h. This function is used to access all variables present in the driver, e.g., the board base address, interrupt or DMA channels from the configuration file 7k.cfg, or EGA parameters.

Example: Listing A-1

k7_bi

Block input to master or slave QPDM.

```
void    k7_bi(count, Buffer);
int     count, Buffer;
```

Block input, 4 bits. The activity bits decide between the master or slave QPDM:

Call k7_act(0x0f) to perform block input to the master QPDM. Call k7_act(0xf0) to perform block input to the slave QPDM.

k7_bom

Block output from master QPDM.

```
void    k7_bom(count, Buffer);
int     count, Buffer;
```

Block output, 4 bits.

Example: Listing A-2

k7_bos

Block output from slave QPDM.

```
void    k7_bos(count, Buffer);
int     count, Buffer;
```

Block output, 4 bits.

Example: Listing A-2

k7_p4to8

Buffer conversion.

void p4to8(*masterBuffer, slaveBuffer, count, image8Buffer*);
int *masterBuffer, slaveBuffer, count, image8Buffer;*

Converts the two 4-bits/pixel buffers from the master and slave QPDM to one buffer with 8 bits/pixel.

k7_bi8

8-bit block input.

void k7_bi8(*count, Buffer*);
int *count, Buffer;*

This function performs a block input with 8 bits/pixel. It is identical to k7_bi of the earlier software version.

Block output can only be done using two buffers, one containing 4 bits/pixel from the master QPDM and the other containing 4 bits/pixel from the slave QPDM. These buffers can be used for the k7_bi function to perform a block input--the activity bits decide if the buffer is transferred to the master or to the slave QPDM. Use this method if you want to make a backup of the screen image without using the image data as input for other purposes.

If you want to access the image data, it is best to convert the two buffers to a byte stream with 8 bits/pixel. Use the k7_p4to8 function to do this.

Set the *bis* and *bos* parameters (of the k7_ibl and k7_obl functions) to 4 when using k7_bi8.

Example: Listing A-3

k7_seldev

Selects one of multiple (independent) K7000CB boards.

```
void    k7_seldev(idev);
int     idev;
```

This function supports multiple K7000CB boards (independent, not cascaded) driving multiple displays. The parameter *idev* (0,1,...max-1) selects a particular board. Note that the boards must be configured in sequential address space, e.g.,

board # 0	I/O space 300h
board # 1	I/O space 320h
board # 2	I/O space 340h
board # 3	I/O space 360h

Note: When initializing the system, call the k7_ini function once, but call k7_res once for each board. For example:

```
k7_ini("\\7k.cfg");
for (i=0; i<nboard; i++)
{       /* loop over all boards */
        k7_seldev(i);
        k7_res();
        k7_fntld();
        k7_usrlut("default.lut");
}
```

k7_filout

Supports QPDM FIFO to a DOS file.

```
void    k7_filout(handle);
int     handle;
```

This function supports on-line QPDM FIFO command level output to a DOS file. The parameter *handle* is the file handler, which is previously opened via the "open" function (in MS-C) or the DOS interrupt 3dh for file open. Setting *handle* = -1 disables the log file function.

Example: Listing A-4

Intentionally Blank

Listing A-1: Access 7k.sys Parameters via Pointers

```
#include "vram.h"

dspparams()                                    /* display 7k.sys params*/
{
     struct P {                                 /* param ptr structure */
         QSTAT *qstat;                      /* QPDM status variables */
         char  *entryp;
         struct K7_CNF *k7_cnf;      /* configuration (7k.dat file) */
         char *modes;
         EGAWIN *egawin;                          /* EGA parameters */
     } pts;

     k7_getpt (&pts);                                /* pointer array */
     printf ("QPDM driver internal variables are:\n\
         base_addr[hex]:  %x\n\
         dma_data:       %d\n\
         dma_fifo:       %d\n\
         interrupt-#:    %d\n",
         pts.k7_cnf->BaseAddress,
         pts.k7_cnf->DmaChannelFifo,
         pts.k7_cnf->DmaChannello,pts.k7_cnf->Interrupt);
     printf ("EGA driver internal variables are:\n\
         x_start:          %d\n\
         y_start:          %d\n\
         char_width:       %d\n\
         char_height:      %d\n\
         chars_per_line:   %d\n\
         # of lines:       %d\n",
         pts.egawin->x_start,pts.egawin->y_start,
         pts.egawin->charwid,pts.egawin->charheight,
         pts.egawin->nword/pts.egawin->nline,
         pts.egawin->nline);
}                                                  /* end of example */
```

Listing A-2: Block I/O Sample #1

This sample program works only with the QPDM Rev C chip.

Each QPDM of an 8-bit system must be read separately (see k7_bom and k7_bos for block output). The data from each QPDM is packed in a "master.dat" and "slave.dat" file.

Input block works with "k7_bi" . The destination of each data stream is determined by "k7_act" (setting activity bits).

```
#include "kon7k.h"
#include <stdio.h>
#include <io.h>
#include <fcntl.h>

FILE *file1, *file2 ;
int handle1, handle2 ;

unsigned short configtab[32] ;
unsigned short iv[19400] ;
unsigned short byteread ;
unsigned short i, j, imax, y=0;
char  dummy;

main ()
{
      ini();

      for (i=0; i<=159; i++)  {
          k7_col (i);
          k7_rct (i*4,0,i*4+3,419)                /* draw rectangles */;
      }
```

Listing A-2, continued

```
k7_blk (640,30);                    /* output block to file on disk */
file1 = fopen ("master.dat","w+b") ;
for (y=0; y<420;) {
    k7_obl (1, 0, y, 4);
    k7_bom (4800, iv);              /* read first QPDM (master) */
    y += 30 ;
    fwrite( (char *) iv, sizeof(unsigned short), 4800, file1);
}
fclose (file1);

file2 = fopen ("slave.dat","w+b") ;
for (y=0; y<420;) {
    k7_obl (1, 0, y, 4);
    k7_bos (4800, iv);              /* read 2nd QPDM (slave) */
    y += 30 ;
    fwrite( (char *) iv, sizeof(unsigned short), 4800, file2);
}
fclose (file2);
k7_col (4);
k7_txt (150,450,"OUTPUT BLOCK source image");


k7_act (0xff) ;                     /* zoom part of source image */
k7_col (0xff) ;
k7_blk (160,105);
k7_xfm (0,0,0,0,640,0,4,4);
k7_col (4);
k7_txt (900,450,"OUTPUT BLOCK zoom-image") ;
```

Listing A-2, continued

```
        byteread = 9600;              /* input block from file on disk */
        k7_col (4);
        k7_txt (150,950,"INPUT BLOCK destination-image") ;
        k7_blk (640,30);
        handle1 = open ("master.dat",O_BINARY);
        k7_act (0x0f);                /* enables writing to first QPDM */
        for (y=0; y<420;)  {
            read (handle1, (char * )iv, byteread);
            k7_ibl (1, 0, y+500, 4);
            k7_bi (4800, iv);                    /* writes to QPDM */
            y += 30;
        }
        close (handle1);

        handle2 = open ("slave.dat",O_BINARY);
        k7_act (0xf0);                /* enables writing to 2nd QPDM */
        for (y=0; y<420;)  {
            read (handle2, (char * )iv, byteread);
            k7_ibl (1, 0, y+500 , 4);
            k7_bi (4800, iv);                    /* writes to QPDM */
            y += 30;
        }
        close (handle2);

        k7_act (0xff);               /* zoom part of input block image */
        k7_col (0xff) ;
        k7_blk (160,105);
        k7_xfm (0,0,0,0,640,500,4,4);
        k7_col (4);
        k7_txt (900,950,"INPUT BLOCK zoom-image");
}
```

Listing A-2, continued

```
ini ()                                          /* initialize board */
{
short i, err;
int m;

    if (err = k7_ini ("7k.cfg"))
        printf ("ini error #%d\n",err);
    if (err = k7_res())
        printf ("res error #%d\n",err);

    k7_enq (configtab);
    printf ("Library revision %d.%d\n",
      configtab[6],configtab[7]);
    printf ("Screen size x=%d y=%d\n",
      configtab[8],configtab[9]);
    ega_color (1);
    ega_lut ();
    ega_set (0);

    k7_sp (1) ;
    k7_am (ABSOLUTE);
    k7_act (0xff);
    k7_wm (LZERO);
    k7_rct (0,0, configtab[8], configtab[9]);
    k7_wm (GSET);
}
```

Listing A-3: Block I/O Sample #2

```
#include <stdio.h>
#include "kon7k.h"
#include "vram.h"
#include <malloc.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys\types.h>
#include <sys\stat.h>

#define xres     512
#define yres      20

char    Buffer [yres][xres];
char    mBuffer [xres*yres/2];
char    sBuffer [xres*yres/2];

main (argc, argv)

int   argc;
char      *argv[];

{
      int  err;
      int  i, j;

for (i=0;i<xres;i++)
      for (j=0;j<yres;j++)
            Buffer [j][i] = i;
```

Listing A-3, continued

```
if (err=k7_ini ("7k.cfg"))                                      /* init driver */
      { printf ("k7_ini: error #%d\n",err); }

      k7_am (ABSOLUTE);
      k7_act (0xff);
      printf ("bi8 \n");
      k7_blk (xres,yres);
      k7_ibl (1,100,100,4);
      k7_bi8 (xres*yres/4, Buffer);
      printf ("bom / bos \n");
      k7_blk (xres,yres);
      k7_obl (1,100,100,4);
      k7_bom (xres*yres/4, mBuffer);

      k7_blk (xres,yres);
      k7_obl (1,100,100,4);
      k7_bos (xres*yres/4, sBuffer);

      printf ("convert in 8 bits/pixel: k7_p4to8 \n");
      k7_p4to8 (mBuffer, sBuffer, xres*yres, Buffer);

      printf ("k7_bi8 \n");
      k7_act (0xff);
      k7_blk (xres, yres);
      k7_ibl (1,100,700,4);
      k7_bi8 (xres*yres/4, Buffer);
}
```

Listing A-4: File Handler Demonstration

```
int handle;

                                                /* open log file */
handle = open ("log",O_CREATE | O_BINARY);
k7_filout (handle);              /* enable QPDM log output */
for (i=0;i<100;i++)  {
      k7_col (i+1);             /* write to QPDM and log file */
      k7_lin (0,i,1000,i);
}
close (handle);                               /* close log file */
k7_filout (-1);                    /* disable log file output */
```