The following pages show the commands and data that is used to communicate with the CubeRevo 9000 front panel.
The front panel is connected using a serial connection (ttyAS0) to the microprocessor.

Serial communication parameters are: ttyAS0, 9600 baud

IMPORTANT NOTE:
The information in this document is NOT necessarily correct and is a work in progress.

Below is a sample application that someone can use to communicates with the front processor.
Note that the example is just for illustration purposes.
Without (or with incorrect) parameters the application is in 'scan' mode and waits max. 5 seconds for data.
This data can be from the remote control or frontpanel keys.
With parameters, the data is interpreted as hexadecimal values, which are send to the front panel
(as a 5 byte packet with unset data being $00). A scan of max 100 ms for returned data is included.

```
//{****************************************************************************}
//{ FileName.............: fptest.c                                             }
//{ Project..............:                                                      }
//{ Author(s)............: M.Majoor                                             }
//{ Version..............: 1.00                                                 }
//{----------------------------------------------------------------------------}
//{  Front processor (micom) test application                                  }
//{----------------------------------------------------------------------------}
//{                                                                            }
//{ Version   Date    Comment                                                  }
//{  1.00   20100619  - Initial release                                        }
//{****************************************************************************}

#include <stdint.h>

#include <linux/input.h>

#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

#include <poll.h>
#include <termios.h>


int main (int argc, char* argv[])
{

  struct termios options;

  int i;
  int inbuf;
  char Data[256];
  int ready;
  struct pollfd polldata[1];
  int day, month, year;
  int hours, minutes, seconds;

  int scanmode = 0;
  int timeout;

  int params[10];
  char *converted;
```

```
for (i=0; i<10; i++)
  params[i] = 0;
// Read parameters
// If invalid parameter revert to 'scan' mode
if (argc > 1) {
  for (i=0; i<(argc-1); i++) {
    params[i] = (int)strtol(argv[i+1], &converted, 16);
    if ((*converted != (char)0) || (params[i] < 0)) {
      scanmode = 1;
    }
  }
}

// Console
polldata[0].fd = open("/dev/ttyAS0", O_RDWR | O_NOCTTY);
polldata[0].events = POLLIN;

fcntl(polldata[0].fd, F_SETFL, FNDELAY);
// Save current options and set to 9600 baud, raw mode
cfsetospeed(&options, B9600);
cfsetispeed(&options, B0);
cfmakeraw(&options);
tcsetattr(polldata[0].fd, TCSANOW, &options);
// 'Purge'
ready = poll(polldata, 1, 10);
if ((ready > 0) && (polldata[0].revents)) {
  read(polldata[0].fd, &Data, 256);
}
if (!scanmode)
{
  // Read micom version
  Data[0] = 0xA5;
  Data[1] = 0x00;
  Data[2] = 0x00;
  Data[3] = 0x00;
  Data[4] = 0x00;
  Data[5] = 0x00;
  write(polldata[0].fd, &Data, 6);
  inbuf = 0;
  do {
    ready = poll(polldata, 1, 10);
    if ((ready > 0) && (polldata[0].revents)) {
      inbuf += read(polldata[0].fd, &Data[inbuf], 256);
    }
  } while (ready > 0);
  if (inbuf == 6) {
    i = 0;
    year  = 0;
    day   = 0;
    month = 0;
    while (i < inbuf) {
      switch (Data[i++] & 0xFF) {
        case 0xE9: // Date
                   day = Data[i++];
                   break;
        case 0xEA: // Month
                   month = Data[i++];
                   break;
        case 0xEB: // Year
                   year = Data[i++];
                   break;
        default:   i++;
                   break;
      }
```

```
  }
  printf("Micom: 20%2.2X-%2.2X-%2.2X\n", year, month, day);
}
else
  printf("Oops, could not read Micom (%d bytes read)\n", inbuf);

printf("= %2.2X %2.2X %2.2X %2.2X =\n", params[0], params[1], params[2], params[3]);
Data[0] = params[0] & 0xFF;
Data[1] = params[1] & 0xFF;
Data[2] = params[2] & 0xFF;
Data[3] = params[3] & 0xFF;
Data[4] = 0x00;
Data[5] = 0x00;
write(polldata[0].fd, &Data, 6);

// 'Purge'
inbuf = 0;
do {
  ready = poll(polldata, 1, 100);
  if ((ready > 0) && (polldata[0].revents)) {
    inbuf += read(polldata[0].fd, &Data[inbuf], 256);
  }
} while (ready > 0);
if (inbuf > 0) {
  for (i=0; i<inbuf; i++) {
    printf("%2.2X ", Data[i] & 0xFF);
  }
  printf("\n");
}
// Read time
Data[0] = 0xA4;
Data[1] = 0x00;
Data[2] = 0x00;
Data[3] = 0x00;
Data[4] = 0x00;
Data[5] = 0x00;
write(polldata[0].fd, &Data, 6);
inbuf = 0;
do {
  ready = poll(polldata, 1, 100);
  if ((ready > 0) && (polldata[0].revents)) {
    inbuf += read(polldata[0].fd, &Data[inbuf], 256);
  }
} while (ready > 0);
if (inbuf == 12) {
  i = 0;
  year    = 0;
  day     = 0;
  month   = 0;
  hours   = 0;
  minutes = 0;
  seconds = 0;
  while (i < inbuf) {
    switch (Data[i++] & 0xFF) {
      case 0xE3: seconds = Data[i++];
                 break;
      case 0xE4: minutes = Data[i++];
                 break;
      case 0xE5: hours = Data[i++];
                 break;
      case 0xE6: day = Data[i++];
                 break;
      case 0xE7: month = Data[i++];
                 break;
```

```
          case 0xE8: year = Data[i++];
                     break;
          default:   i++;
                     break;
        }
      }
      printf("20%2.2X-%2.2X-%2.2X %2.2X:%2.2X:%2.2X\n",
        year, month, day, hours, minutes, seconds);
    }
    else
      printf("Oops, could not read time (%d bytes read)\n", inbuf);
  }

  if (scanmode)
  {
    timeout = 5000; // Initial timeout
    do
    {
      inbuf = 0;
      ready = poll(polldata, 1, timeout);
      if ((ready > 0) && (polldata[0].revents)) {
        timeout = 250;
        inbuf += read(polldata[0].fd, &Data[inbuf], 256);
        for (i=0; i<inbuf; i++)
          printf("%2.2X ", Data[i] & 0xFF);
      }
    } while (ready > 0);
    printf("\n");
  }

  return 0;
}
```

Data received is in 2 byte packets.
The first byte indicates the type of data that follows and the second byte is the actual data.

Byte 0	Byte 1	Values	Description
$E0	**Key**		Remote control key
			See below for key codes
		$FF	Key is released
$E1	**Key (MSB)**		Front panel key (MSB)
$E2	**Key (LSB)**		Front panel key (LSB)
		$0000	Key is released
		$xxx1	Power key pressed
		$xxx2	Menu key pressed
		$xxx4	Exit key pressed
		$xxx8	File key pressed
		$xx1x	OK key pressed
		$xx2x	Left key pressed
		$xx4x	Right key pressed
		$xx8x	Up key pressed
		$x1xx	Down key pressed
			Send order: $E1, $E2
$E3	**Second**		Time (BCD)
$E4	**Minute**		Time (BCD)
$E5	**Hour**		Time (BCD)
$E6	**Day**		Time (BCD)
$E7	**Month**		Time (BCD)
$E8	**Year**		Time (BCD), offset by 2000
			Send order: $E3, $E4, $E5, $E6, $E7,$E8
$E9	**Day**		Micom version day (BCD)
$EA	**Month**		Micom version month (BCD)
$EB	**Year**		Micom version year (BCD), offset by 2000
			Send order: $EB, $EA, $E9
$EC	**Data**		RAM value
$ED	**Status**		Status of alarm/wake up
		$00	No alarm active
		$02	Alarm active
$EE	**$00**		
$EF	**$00**		

$E1 key codes:

$00	Sub picture position in PIP
$01	Swap picture
$02	Sub channel list in PIP
$03	Info banner
$04	TV/Radio
$05	Recall
$06	Electronic program guide
$07	Record
$08	Favorite
$09	Media list
$0A	Switch on/off

$0B	Music list
$0C	Audio mute
$0D	Photo album list
$0E	-not on remote-
$0F	-not on remote-
$10	0
$11	1
$12	2
$13	3
$14	4
$15	5
$16	6
$17	7
$18	8
$19	9
$1A	Navigation up
$1B	Navigation down
$1C	Navigation right
$1D	Navigation left
$1E	Fast reverse
$1F	OK
$20	Playback
$21	Fast forward
$22	Stop
$23	Slow motion
$24	Repeat
$25	Pauze
$26	Menu
$27	Exit
$28	PIP on/off
$29	Book or check mark
$2A	Red
$2B	Blue
$2C	Multifeed
$2D	Green
$2E	Yellow
$2F	Alternate audio
$30	Subtitle
$31	Teletext
$32	Page/program up
$33	Page/program down
$34	Volume up
$35	Volume down
$36	www
$FF	Released

Writing is done using a 5 byte packet.
The first byte indicates the type of command and the others bytes can have additional data.
The last byte of a packet is always $00.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Values	Description
$A0	**$00**	**$00**	**$00**	**$00**		
						See received data ($EF)
$A1	**$00**	**$00**	**$00**	**$00**		Get alarm/wake up status
						See received data ($ED)
$A2	**$00**	**$00**	**$00**	**$00**		
						See received data ($EE)
$A3						Read RAM
	address					Address to read
					$00-$3F	Address range
					$00	Time second (BCD)
					$01	Time minute (BCD)
					$02	Time hour (BCD)
					$03	Time day (BCD)
					$04	Time month (BCD)
					$05	Time month (BCD)
					$06	Time year (BCD)
		$00	**$00**	**$00**		See received data ($EC)
$A4	**$00**	**$00**	**$00**	**$00**		Get time/date
						See received data ($E3..$E8)
$A5	**$00**	**$00**	**$00**	**$00**		Get micom version
						See received data ($E9..$EB)
$A6	**$00**	**$00**	**$00**	**$00**		Get alarm time/date
						See received data ($E3..$E8)
$C0						Set alarm/wake up (date)
	Year					Year (BCD), offset by 2000
		Month				Month (BCD)
			Day			Day (BCD)
				$00		
$C1						Set alarm/wake up (time)
	Hour					Hour (BCD)
		Minute				Minute (BCD)
			OnOff			Set alarm/wake up on/off
					$00	Alarm/wake up off
					<> $00	Alarm/wake up on
				$00		
$C2						Set time (date)
	Year					Year (BCD), offset by 2000
		Month				Month (BCD)
			Day			Day (BCD)
				$00		
$C3						Set brightness level
	Level					Brightness level
					%xxxxxXXX	0-7 (min-max)
		$00	**$00**	**$00**		
$C4						Set time in VFD display
	OnOff					
					%xxxxxxx0	No time displayed in VFD
					%xxxxxxx1	Time displayed in VFD
		$00	**$00**	**$00**		
$C5						Set LED
	OnOff					
					$00	LED off
					<> $00	LED on
		$00	**$00**	**$00**		
$C6						Set LED (slow toggle/blink)
	OnOff					
					%xxxxxxx0	LED last state will be off
					%xxxxxxx1	LED last state will be on
					%1xxxxxxx	Continuous toggle/blink
					%0XXXXXXx	LED toggles/blinks (XXXXXX times to toggle
		$00	**$00**	**$00**		on/off before entering last state)
$C7						Set LED (fast toggle/blink)
	OnOff					
					%xxxxxxx0	LED last state will be off
					%xxxxxxx1	LED last state will be on
					%1xxxxxxx	Continuous toggle/blink
					%0XXXXXXx	LED toggles/blinks (XXXXXX times to toggle
		$00	**$00**	**$00**		on/off before entering last state)
$C8						Set shutdown/reboot/warmstart on/warmstart off
	State					
					$00	Shutdown device
					$01	Reboot device
					$02	Warmstart off (does not reboot)
					$03	Warmstart on (does not reboot)

		$00	**$00**	**$00**		

$C9						Set RAM
	address					Address
		Data				Data
			$00	**$00**		

$CA						Set time
	Hour					Hour (BCD)
		Minute				Minute (BCD)
			Second			Second (BCD)
				$00		

$CC	**$00**	**$00**	**$00**	**$00**		Set fan on

$CD	**$00**	**$00**	**$00**	**$00**		Set fan off

$CE	**$00**	**$00**	**$00**	**$00**		Set RF modulator on

$CF	**$00**	**$00**	**$00**	**$00**		Set RF modulator off

$D0						Set text in buffer (is not yet displayed!)
	Position					Position in buffer
					0-11	Position
		Data				Character to place in buffer (see Sheet_ASCII)
			$00	**$00**		

$D1	**$00**	**$00**	**$00**	**$00**		Display text buffer

$D5	**$00**	**$00**	**$00**	**$00**		Clear text buffer

$D7						Set time mode
	Mode					12h/24h/ mode
					$00	12h mode
					$01	24h mode
		$00	**$00**	**$00**		

$D8						Set segments I (see VFD layout sheet)
	Mode					
					$00	Segment off
					$01	Segment off
		Segment (LSB)	**Segment (MSB)**			
					$0000	RECording (*) Automatically switched
					$0101	TIME on/off when device is shutdown
					$0102	SHIFT based on wake up alarm status
					$0103	Timer (*) Needs to be switched off
					$0104	HD manually after powering up
					$0105	USB
					$0106	Locked
					$0107	Dolby
					$0108	Mute
					$0109	Tu1
					$010A	Tu2
					$010B	MP3
					$010C	Repeating
				$00		

$D9						Set segments II (see VFD layout sheet)
	Mode					Segment state
					$00	Segment off
					$01	Segment on
		Segment (LSB)	**Segment (MSB)**			Segment number
					$0000	Play mode segment NNE
					$0001	Play mode segment NNW
					$0002	Play mode segment ENE
					$0003	Play mode segment WNW
					$0004	Play mode segment circle
					$0100	Play mode segment play
					$0101	Play mode segment ESE
					$0102	Play mode segment WSW
					$0103	Play mode segment SSE
					$0104	Play mode segment SSW
					$0200	SATellite
					$0201	TERestrial
					$0202	FILE
					$0203	TV
					$0204	Radio
					$0300	Standby
					$0501	1080p
					$0502	1080i
					$0503	720p
					$0504	p (of 576)
					$0600	i (of 576)
					$0601	576
					$0602	p (of 480)
					$0603	i (of 480)
					$0604	480
				$00		

$DA	**$00**	**$00**	**$00**	**$00**		Clear segments I/II

Standby	Satellite	Recording	Time	Shift	Timer	HD	USB	Locked	Dolby	Mute	Tuner 1	Tuner 2	MP3	Repeating	1080p
D9:0300	**D9:0200**	**D8:0000**	**D8:0101**	**D8:0102**	**D8:0103**	**D8:0104**	**D8:0105**	**D8:0106**	**D8:0107**	**D8:0108**	**D8:0109**	**D8:010A**	**D8:010B**	**D8:010C**	**D9:0501**

Radio **D9:0204** — Terrestrial **D9:0201**

Play mode

Tv **D9:0203** — File **D9:0202**

D9:0001 **D9:0000**

D9:0003 **D9:0002**

D9:0102 **D9:0101**

D9:0104 **D9:0103**

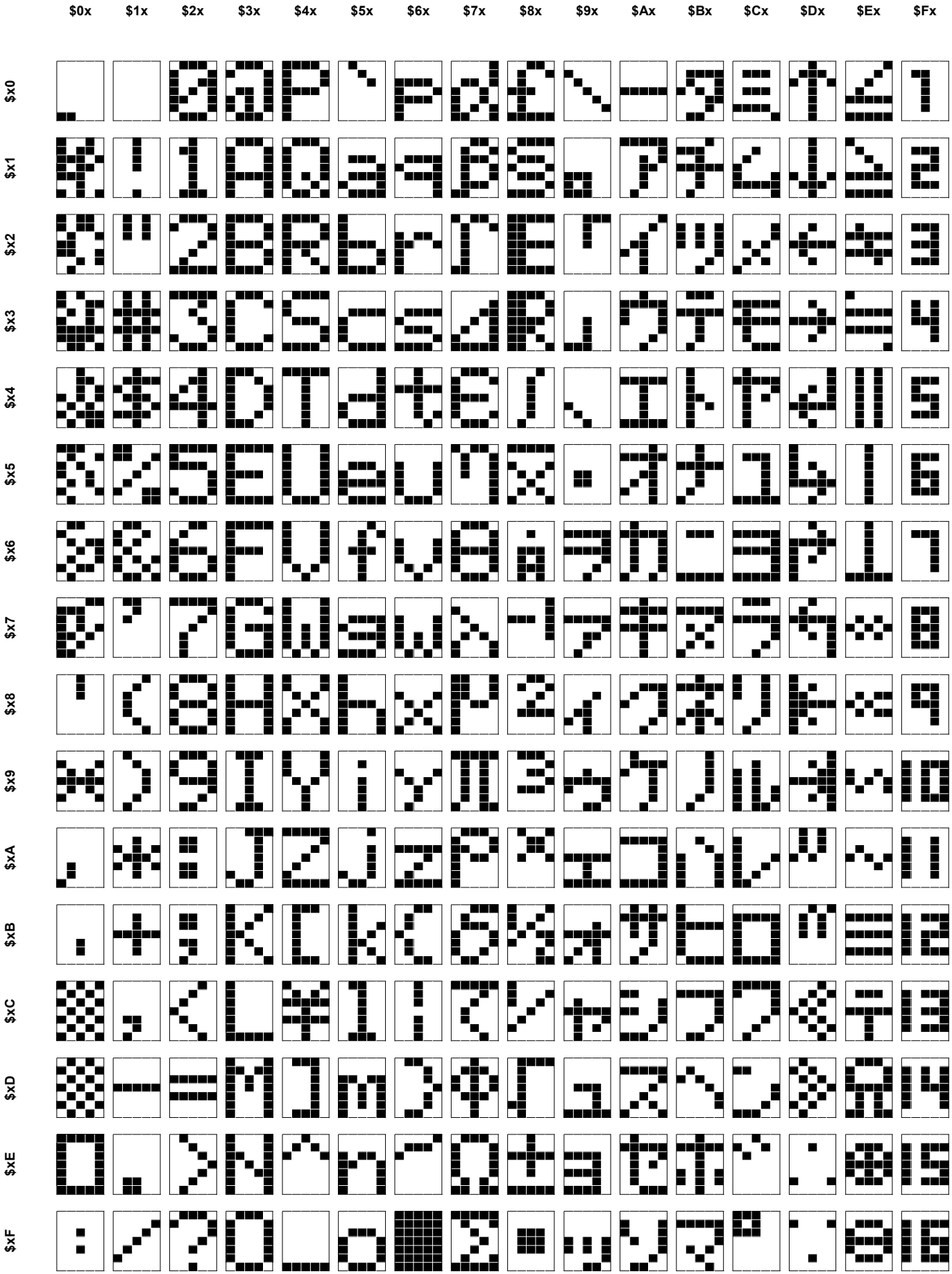
D9:0004 **D9:0100**

1080i **D9:0502**

720p **D9:0503**

576	Interlaced	Progressive
D9:0601	**D9:0600**	**D9:0504**

480	Interlaced	Progressive
D9:0604	**D9:0603**	**D9:0602**


$0x
$1x
$2x
$3x
$4x
$5x
$6x
$7x
$8x
$9x
$Ax
$Bx
$Cx
$Dx
$Ex
$Fx
$x0
$x1
$x2
$x3
$x4
$x5
$x6
$x7
$x8
$x9
$xA
$xB
$xC
$xD
$xE
$xF