Displaying the attribute characters on your Green Screen

Tech tip courtesy of Douglas Handy

This tip appeared on the MIDRANGE-L list recently and, since there was significant interest at the November, 2004 general meeting, we asked Doug for permission to add it to our T & T pages. So here it is . . .

You may remember that, on the old 5251 workstations, there was a “test” switch that showed each attribute byte as its hexadecimal value. There are a (very) few displays in use today that have that capability. I can think of no PC-based emulators that can do it. Yet, it is a nice debugging tool when developing display panels. Well, now you can do it on ANY emulator – read on.

The 5250 data stream has information about things to display, how to allow input and other features. Each field is introduced (and ended) by a position on the screen with an attribute. These attribute bytes are hexadecimal values between x'20' and x'3F' and control what you see, how it looks and even whether or not you see it (e.g., non-display fields). There are, therefore, 32 possible values and their interpretation by the display hardware/emulator depends upon whether you are in “color” or “good-old-green” mode.

The following table (from IBM's Information Center) shows the values and their meanings:

Hex	Limited Color	Full Color
20	Normal	Green
21	Reverse image	Green, reverse image
22	High intensity	White
23	High intensity, reverse image	White, reverse image
24	Underscore	Green, underscore
25	Underscore, reverse image	Green, underscore, reverse image
26	Underscore, high intensity	White, underscore
27	Nondisplay	Nondisplay
28	Blink	Red
29	Blink, reverse image	Red, reverse image
2A	Blink, high intensity	Red, high intensity
2B	Blink, high intensity, reverse image	Red, high intensity, reverse image
2C	Blink, underscore	Red, underscore
2D	Blink, underscore, reverse image	Red, underscore, reverse image
2E	Blink, underscore, high intensity	Red, underscore, blink
2F	Nondisplay	Nondisplay

Hex	Limited Color	Full Color
30	Column separator	Turquoise, column separator
31	Reverse image, column separator	Turquoise, column separator, reverse image
32	High intensity, column separator	Yellow, column separator
33	High intensity, reverse image, column separator	White, reverse image, column separator
34	Underscore, column separator	Turquoise, underscore, column separator
35	Underscore, reverse image, column separator	Turquoise, underscore, reverse image, column separator
36	Underscore, high intensity, column separator	Yellow, underscore, column separator
37	Nondisplay	Nondisplay
38	Blink, column separator	Pink
39	Blink, reverse image, column separator	Pink, reverse image
3A	Blink, high intensity, column separator	Blue
3B	Blink, high intensity, reverse image, column separator	Blue, reverse image
3C	Blink, underscore, column separator	Pink, underscore
3D	Blink, underscore, reverse image, column separator	Pink, underscore, reverse image
3E	Blink, underscore, high intensity, column separator	Blue, underscore
3F	Nondisplay	Nondisplay

Here's the text of Doug's posting and the full code for the RPGLE program. Compile it at V5R1M0 or later because of the free-form calcs. It works!

My solution was conceptually similar, but used DSM instead of USRDFN data streams. In addition, instead of clearing the unit and displaying the image as one output field, I leave the input fields intact and just overwrite the attribute bytes. Then instead of simply waiting for an AID key and exiting, I loop while Enter is pressed, displaying the hex value of the cursor location until some other AID key is pressed (eg F3 or F12).

I wasn't sure I should put a 200 line source directly in a reply, but for the sake of comparison, here is a RPG alternative to the CL program. Once upon a time I used USRDFN, but now consider DSM much more readable.

Here is my source, which I called DspDspAtr:

```
     H Option( *SrcStmt : *NoDebugIO )
     H DftActGrp( *No )
     H ActGrp( *Caller )
     H BndDir( 'QC2LE' )

      * Display display attributes
```

```
 * Use SETATNPGM DSPDSPATR then use ATTN key to invoke this program.
 * The current Screen will have all display attributes replaced by
 * a @ character.  Move the cursor and press Enter to have the hex
 * value of that position displayed.  Use any Fx key to exit.

 * Copyright 2004 Douglas Handy.
 * Permission is granted to distribute freely; all other rights
 * are reserved.

 * Stand-alone variables used
D BegRow          S             10I 0
D BegCol          S             10I 0
D Rows            S             10I 0
D Cols            S             10I 0
D R               S             10I 0
D C               S             10I 0
D Hex             S              2

D CmdBuf          S             10I 0
D InpHnd          S             10I 0
D BytRead         S             10I 0

D ScrImg          S           3564
D ScrImgPtr       S               *   Inz( *Null )
D ScrBytePtr      S               *   Inz( *Null )
D ScrByte         S              1    Based( ScrBytePtr )

D InpDtaPtr       S               *   Inz( *Null )
D InpDta          DS          3564    Based( InpDtaPtr )
D  InpCsrRow                     3U 0
D  InpCsrCol                     3U 0
D  InpAID                        1

 * Convert character string to hex string (eg ABC to C1C2C3)
D CvtToHex        PR                  ExtProc( 'cvthc' )
D  Hex                        2048    Options( *Varsize )
D  Char                       1024    Options( *Varsize )
D  LenSrc                       10I 0 Value

 * Copy a block of memory (operands should not overlap)
D memcpy          PR              *   ExtProc( '__memcpy' )
D  Target                         *   Value
D  Source                         *   Value
D  Length                       10U 0 Value

 * Standard API error code DS
D ApiErrCode      DS
D  ErrBytPrv                     9B 0 Inz( %size( ApiErrCode ) )
D  ErrBytAvl                     9B 0 Inz( 0 )
D  ErrMsgID                      7
D  ErrResv                       1
D  ErrMsgDta                    80

 * Retrieve Screen dimensions of current mode (not capability).
D RtvScrDim       PR            10I 0 ExtProc( 'QsnRtvScrDim' )
D  Rows                         10I 0
D  Cols                         10I 0
D  EnvHnd                       10I 0 Options( *Omit ) Const
D  ErrorDS                            Options( *Omit ) Like( ApiErrCode )

 * Clear buffer.
D ClrBuf          PR            10I 0 ExtProc( 'QsnClrBuf' )
D  CmdBuf                       10I 0 Options( *Omit ) Const
```

```
D  ErrorDS                            Options( *Omit ) Like( ApiErrCode )

 * Create command buffer.
D CrtCmdBuf       PR            10I 0 ExtProc( 'QsnCrtCmdBuf' )
D  InitSize                     10I 0 Const
D  IncrAmt                      10I 0 Options( *Omit ) Const
D  MaxSize                      10I 0 Options( *Omit ) Const
D  CmdBuf                       10I 0 Options( *Omit ) Const
D  ErrorDS                            Options( *Omit ) Like( ApiErrCode )

 * Create input buffer.
D CrtInpBuf       PR            10I 0 ExtProc( 'QsnCrtInpBuf' )
D  InitSize                     10I 0 Const
D  IncrAmt                      10I 0 Options( *Omit ) Const
D  MaxSize                      10I 0 Options( *Omit ) Const
D  InpBuf                       10I 0 Options( *Omit )
D  ErrorDS                            Options( *Omit ) Like( ApiErrCode )

 * Delete buffer.
D DltBuf          PR            10I 0 ExtProc( 'QsnDltBuf' )
D  BufHnd                       10I 0 Const
D  ErrorDS                            Options( *Omit ) Like( ApiErrCode )

 * Read Screen (without waiting for an AID key).
D ReadScr         PR            10I 0 ExtProc( 'QsnReadScr' )
D  NbrByt                       10I 0 Options( *Omit )
D  InpBuf                       10I 0 Options( *Omit ) Const
D  CmdBuf                       10I 0 Options( *Omit ) Const
D  EnvHnd                       10I 0 Options( *Omit ) Const
D  ErrorDS                            Options( *Omit ) Like( ApiErrCode )

 * Retrieve pointer to data in input buffer.
D RtvDta          PR              *   ExtProc( 'QsnRtvDta' )
D  InpBuf                       10I 0 Const
D  InpDtaPtr                      *   Options( *Omit )
D  ErrorDS                            Options( *Omit ) Like( ApiErrCode )

 * Read input fields.
D ReadInp         PR            10I 0 ExtProc( 'QsnReadInp' )
D  CCByte1                       1    Const
D  CCByte2                       1    Const
D  NbrFldByt                    10I 0 Options( *Omit )
D  InpBuf                       10I 0 Options( *Omit ) Const
D  CmdBuf                       10I 0 Options( *Omit ) Const
D  EnvHnd                       10I 0 Options( *Omit ) Const
D  ErrorDS                            Options( *Omit ) Like( ApiErrCode )

 * Get cursor address (does not wait for AID key).
D GetCsrAdr       PR            10I 0 ExtProc( 'QsnGetCsrAdr' )
D  CsrRow                       10I 0 Options( *Omit )
D  CsrCol                       10I 0 Options( *Omit )
D  EnvHnd                       10I 0 Options( *Omit ) Const
D  ErrorDS                            Options( *Omit ) Like( ApiErrCode )

 * Set cursor address.
D SetCsrAdr       PR            10I 0 ExtProc( 'QsnSetCsrAdr' )
D  FldID                        10I 0 Options( *Omit ) Const
D  CsrRow                       10I 0 Options( *Omit ) Const
D  CsrCol                       10I 0 Options( *Omit ) Const
D  CmdBuf                       10I 0 Options( *Omit ) Const
D  EnvHnd                       10I 0 Options( *Omit ) Const
D  ErrorDS                            Options( *Omit ) Like( ApiErrCode )
```

```
 * Write data.
D WrtDta          PR            10I 0 ExtProc( 'QsnWrtDta' )
D  Data                       3600    Const
D  DataLen                      10I 0 Const
D  FldID                        10I 0 Options( *Omit ) Const
D  Row                          10I 0 Options( *Omit ) Const
D  Col                          10I 0 Options( *Omit ) Const
D  StrMonoAtr                    1    Options( *Omit ) Const
D  EndMonoAtr                    1    Options( *Omit ) Const
D  StrClrAtr                     1    Options( *Omit ) Const
D  EndClrAtr                     1    Options( *Omit ) Const
D  CmdBuf                       10I 0 Options( *Omit ) Const
D  EnvHnd                       10I 0 Options( *Omit ) Const
D  ErrorDS                            Options( *Omit ) Like( ApiErrCode )

C/Free

  // Get display size and save current contents of Screen image
  RtvScrDim( Rows: Cols: *Omit: *Omit );
  GetCsrAdr( BegRow: BegCol: *Omit: *Omit );
  InpHnd     = CrtInpBuf( %size( ScrImg ): *Omit: *Omit: *Omit: *Omit );
  BytRead    = ReadScr( *Omit: InpHnd: *Omit: *Omit: *Omit );
  InpDtaPtr  = RtvDta( InpHnd: *Omit: *Omit );
  ScrImgPtr  = %addr( ScrImg );
  memcpy( ScrImgPtr : InpDtaPtr: BytRead );

  // Create command buffer with an output command to replace
  // each display attribute byte with a @ character, except
  // for the attribute at row/col 1,1 because overlaying it
  // affects at least some emulators
  CrtCmdBuf( 1024: 1024: 6192: CmdBuf: *Omit );
  ScrBytePtr = %addr( ScrImg );

  For R = 1 to Rows;
    For C = 1 to Cols;
      If ScrByte >= x'20' and ScrByte <= x'3F';
        If not ( R = 1 and C = 1 );
          WrtDta( '@': 1: 0: R: C: *Omit: *Omit: *Omit: *Omit:
                  CmdBuf: *Omit: *Omit );
        Endif;
      Endif;
      ScrBytePtr = ScrBytePtr + 1;
    Endfor;
  Endfor;

  // Output cmd buffer to display and wait for AID key
  SetCsrAdr( *Omit: BegRow: BegCol: CmdBuf: *Omit: *Omit );
  ReadInp( x'20': x'40': BytRead: InpHnd: CmdBuf: *Omit: *Omit );
  InpDtaPtr  = RtvDta( InpHnd: *Omit: *Omit );

  // Show hex contents of cursor position until Enter not pressed
  Dou InpAID <> x'F1';
    ClrBuf( CmdBuf: *Omit );
    ScrBytePtr = ScrImgPtr + ( ( InpCsrRow - 1 ) * Cols ) + InpCsrCol - 1;
    CvtToHex( Hex: ScrByte: 2 );
    WrtDta( Hex: 2: 0: Rows: Cols-1: x'22': *Omit: x'22': *Omit:
            CmdBuf: *Omit: *Omit );
    SetCsrAdr( *Omit: InpCsrRow: InpCsrCol: CmdBuf: *Omit: *Omit );
    ReadInp( x'20': x'40': BytRead: InpHnd: CmdBuf: *Omit: *Omit );
    InpDtaPtr  = RtvDta( InpHnd: *Omit: *Omit );
  Enddo;

  // Delete DSM buffers and end program
  DltBuf( CmdBuf: *Omit );
  DltBuf( InpHnd: *Omit );
  *InLR = *On;
 /End-free
```