```
// Format of the mouse cursor header.
typedef struct      tagCURSORHEADER
        {
        WORD  wHotSpotX;
        WORD  wHotSpotY;
        WORD  wExtentX;
        WORD  wExtentY;
        WORD  wFunnyNumber;
        BYTE   ucNumberOfPlanes;
        BYTE   ucBitsPerPixel;
        }       CURSORHEADER;

typedef CURSORHEADER FAR * LPCURSORHEADER;

// This code may be added to GENERIC's MainWndProc message switch.
case WM_KEYDOWN:
   switch ( wParam )
        {
        case VK_RETURN:
           {
           // Swap the system Arrow and HourGlass cursors.
           // These cursors can be manipulated in memory because they
           //        are considered non-discardable and will never be
           //        reloaded from disk.
           // Several of the function calls should really be checked
           //        for error returns.

           HCURSOR        hCursorARROW, hCursorHOUR;
           LPSTR          lpGMemCursorARROW, lpGMemCursorHOUR;
           HANDLE         hMemTemp;
           LPSTR          lpGMemTemp, lpGMemSrc, lpGMemDst;
           LPCURSORHEADER  lpCursorHdr;
           WORD           wExtX, wExtY, wCursorSize, i;

           // Get handles to the system Arrow & Hour Glass cursors.
           hCursorARROW = LoadCursor( NULL, IDC_ARROW );
           hCursorHOUR  = LoadCursor( NULL, IDC_WAIT  );

           // Lock the cursor handles down.
           lpGMemCursorARROW = GlobalLock( hCursorARROW );
           lpGMemCursorHOUR  = GlobalLock( hCursorHOUR  );

           // Cast the structure pointer to one of the cursor address.
           lpCursorHdr = (LPCURSORHEADER)lpGMemCursorARROW;

           // Calculate the size of the cursors.
           wExtX = lpCursorHdr->wExtentX;
           wExtY = lpCursorHdr->wExtentY;
           wCursorSize = sizeof( CURSORHEADER ) +
                   2 * ( ( wExtX / 8 ) * wExtY );

           // Swap ARROW and HOUR cursors using a temp swap area.
           // Allocate some temporary swap space.
           hMemTemp = GlobalAlloc( GMEM_MOVEABLE | GMEM_ZEROINIT ,
                   wCursorSize );
           lpGMemTemp = GlobalLock( hMemTemp );

           // Copy ARROW to temp.
```

```
            lpGMemSrc = lpGMemCursorARROW;
            lpGMemDst = lpGMemTemp;
            for ( i = 0;  i < wCursorSize;  i++ )
                  *lpGMemDst++ = *lpGMemSrc++;

            // Copy HOUR to ARROW.
            lpGMemSrc = lpGMemCursorHOUR;
            lpGMemDst = lpGMemCursorARROW;
            for ( i = 0;  i < wCursorSize;  i++ )
                  *lpGMemDst++ = *lpGMemSrc++;

            // Copy Temp to HOUR.
            lpGMemSrc = lpGMemTemp;
            lpGMemDst = lpGMemCursorHOUR;
            for ( i = 0;  i < wCursorSize;  i++ )
                  *lpGMemDst++ = *lpGMemSrc++;

            // Unlock and Free the temp swap.
            GlobalUnlock( hMemTemp );
            GlobalFree( hMemTemp );

            // Unlock the system cursors.
            GlobalUnlock( hCursorARROW );
            GlobalUnlock( hCursorHOUR );

            // Force a repaint of the cursor.
            ShowCursor( FALSE );
            ShowCursor( TRUE );
            break;
            }
      }
break;
```