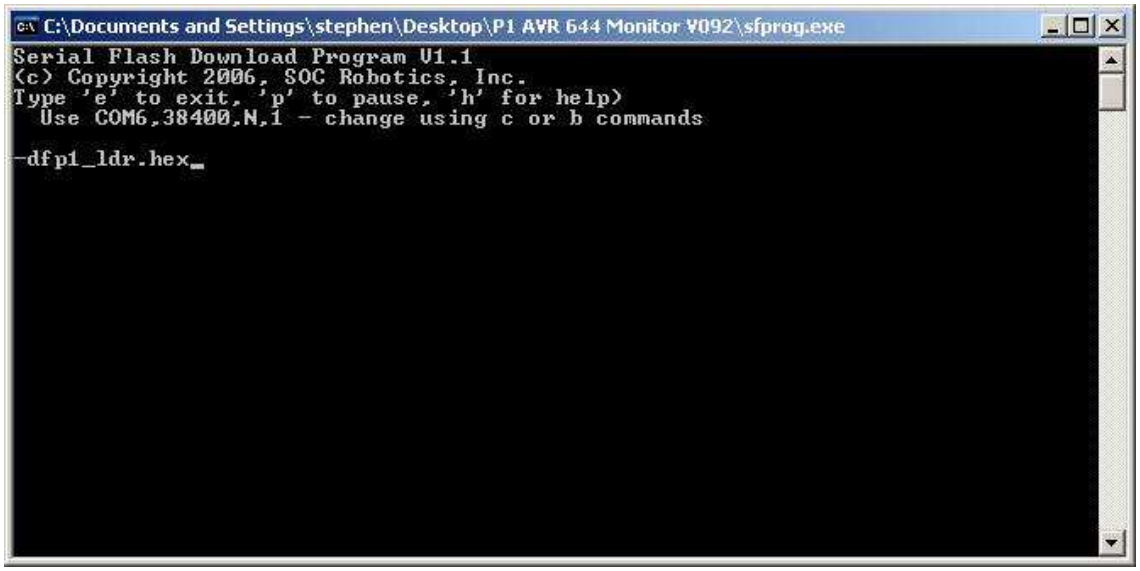
APPLICATION NOTE

CREATING A P1 BOOTABLE IMAGE USING VISUAL DSP

The P1 Boot Monitor (P1BM) contains functions for reading/writing memory (SDRAM and Serial Flash), displaying system status, initializing peripherals and loading Intel Hex files. The P1BM image is about 51Kbyte so it doesn't fit in the Blackfin's instruction cache (32K). The VisualDSP linking loader allows the programmer to place code in off chip devices such as SDRAM. The Blackfin memory manager then automatically loads the off chip program segments into on chip instruction cache during program execution. The P1BM was developed using VisualDSP and includes all source and build project files. For ease of development VisualDSP automatically includes key initialization code when an attached JTAG emulator is used for debugging (we use an Analog Devices ADZS-UPUSB-ICE Emulator for in house development). In order to create an executable image that doesn't rely on an emulator the target boot image must include initialization code to make sure critical peripherals are properly initialized before the main body of code is loaded.

When the P1 boots from an external Serial Flash chip it loads an initial section of code that initializes the main system clock and SDRAM registers. The programmer is responsible for creating this initial startup code. Included with the P1 SDK distribution is a startup file called **`P1_startup.asm`**. This file contains the code to initialize the system clock and SDRAM registers. **`"P1 Startup.dpj"`** is a VisualDSP project file that converts **`P1_startup.asm`** into **`P1s.dxe`**. **`P1s.dxe`** is then combined with the P1Boot Monitor program image .dxe file to form the P1 Target Intel Hex boot image.

The **`ELFLoader.exe`** is a utility for combining multiple Blackfin compiled image files (.dxe) into a single downloadable Intel Hex file (.hex). The .hex file is then loaded into P1 Serial Flash using an AVR Boot Monitor Hex file load command. A special program called **`sfprog.exe`** is used to send the hex file to the AVR from a host PC. After sfprog.exe starts type the command below:



Included with the P1 SDK distribution is a batch file called **`p1init.bat`** containing the line:

```
>elfloader "P1 Blackfin Monitor" -proc ADSP-BF532 -init P1s.dxe -f hex -b spi -o
p1_ldr.hex
```

where **"P1 Blackfin Monitor"** is the P1BM, **P1s.dxe** is the P1_startup.asm image, **spi** selects Mode 3 Serial Flash boot load mode and **p1_ldr.hex** is the output Intel Hex file ready for downloading to the P1.

The ELFloader creates a sequence of code segments that the Blackfin on chip boot code interprets as memory location directives – so **P1_startup.asm** is loaded first and run so SDRAM and the system clock are initialized properly. The Blackfin then continues to load the rest of P1BM into on chip cache and external SDRAM. After the P1BM is loaded execution starts at main(). If the SDRAM is not configured or the SDRAM and CLOCK init routines are left in the P1BM code the boot will fail.

The **Init_Blackfin(void)** routine contains the initialization calls for SDRAM and Clock setrup– comment this code out to create a Bootable image.

```
//  Setup Blackfin and initialize all peripherals
void Init_Blackfin(void)
{
        //Initialize System Configuration Register
//      sysreg_write(reg_SYSCFG, 0x32);

        //  Initialize periperals and system components
        //  Note:  For Flashed versions do not reinitialize PLL, SDRAM and Timers
//      Init_PLL();
//      Init_SDRAM();
//      Init_Timers();
        //  Start code again
        Init_EBIU();        //  Activate APS12 Ethernet Adapter CS
//      Init_PPI_CM64();    //  Make sure Video Port PPI and PF8-15 are inputs
//      Init_UART(57600);
        Init_UART(115200);
//      Init_UART(9600);

        Init_Interrupts();
        Init_RealTimeClock();
        Init_MasterSPI();  // Default setting - if the AVR needs to access Serial Flash
                                        // then the Blackfin must relinguish control

        // Watchdog timer setup
        InitWatchdog();

    // Setup constants and variables
        programmode = LBlackfinSDRAM;

        //  Output startup message on UART port
        SignOnMessage();


        //  Output Ethernet and related configuration information
//      InternetSettings();

        //  Set Intel Hex extended address variable to zero and
        //  initialize Flash programming parameters - SPI Master
        //  Initialize Intel hex download parameters for Flash loading
        ResetFlashVars();
        intelhexecho = 0x01;
        intelhexprintmode = SHORTMODE;
        Init_Flash();

        // Check Flash Type(s) and set size parameters
        //  512K,1M or 2M per Flash - sets the various
        //  parameters that interact with the Flash
        CheckFlash();

        //  Initialize the communication protocol to allow communication
        //  with the AVR
        if(Init_AVR()==-1)  puts("AVR not communicating - TBD\n\r");

        //  Display command prompt
```

```
        //    Note:  Blackfin command prompt is '>' while AVR prompt is ':'
        putchar('>');

}
```

P1_Startup.asm is below:

```
#include <defBF532.h>
.section program;
/*********************************************************/
/**********************Pre-Init Section**********************/

        [--SP] = ASTAT;   /* Stack Pointer (SP) is set to the end of */
        [--SP] = RETS;    /* scratchpad memory (0xFFB00FFC) */
        [--SP] = (r7:0);  /* by the on-chip boot ROM */
        [--SP] = (p5:0);
        [--SP] = I0;[--SP] = I1;[--SP] = I2;[--SP] = I3;
        [--SP] = B0;[--SP] = B1;[--SP] = B2;[--SP] = B3;
        [--SP] = M0;[--SP] = M1;[--SP] = M2;[--SP] = M3;
        [--SP] = L0;[--SP] = L1;[--SP] = L2;[--SP] = L3;

/*********************************************************/
/****** Init Code Section *********************************/

/****** PLL Setup *************/
//  Two master clock rates - fast SDRAM and slow SDRAM
//  Some P1's operate at fast SDRAM rates - others don't
Setup_PLL:
        P0.L = lo(PLL_CTL);
        P0.H = hi(PLL_CTL);    // PLL Control Register
//      R0 = 0x3A00(Z);                 //  391.5MHz cclk
//      R0 = 0x1E00(Z);                 //  405.0MHz cclk
        R0 = 0x1C00(Z);                 //  378.0MHz cclk - added March 26, 2006
        W[P0] = R0;

        P0.L = lo(PLL_DIV);
        P0.H = hi(PLL_DIV);    // PPL Divide Register
//      R0 = 0x0016(Z);                 //  130.00MHz sclk
//      R0 = 0x0004(Z);                 //  101.25MHz sclk
        R0 = 0x0003(Z);                 //  126.00MHz sclk - added March 26, 2006
        W[P0] = R0;
        IDLE;
        SSYNC;

/****** SDRAM Setup *************/
Setup_SDRAM:
        P0.L = lo(EBIU_SDRRC);
        P0.H = hi(EBIU_SDRRC);      // SDRAM Refresh Rate Control Register
        R0 = 0x081A(Z);
        W[P0] = R0;
        SSYNC;

        P0.L = lo(EBIU_SDBCTL);
        P0.H = hi(EBIU_SDBCTL);    // SDRAM Memory Bank Control Register
        R0 = 0x0000(Z);
        W[P0] = R0;
        SSYNC;

        P0.L = lo(EBIU_SDGCTL);
        P0.H = hi(EBIU_SDGCTL);    // SDRAM Memory Global Control Register
        R0.H = 0xE008;
        R0.L = 0x8849;
        [P0] = R0;
        SSYNC;

/*********************************************************/
        L3 = [SP++]; L2 = [SP++]; L1 = [SP++]; L0 = [SP++];
        M3 = [SP++]; M2 = [SP++]; M1 = [SP++]; M0 = [SP++];
        B3 = [SP++]; B2 = [SP++]; B1 = [SP++]; B0 = [SP++];
        I3 = [SP++]; I2 = [SP++]; I1 = [SP++]; I0 = [SP++];
        (P5:0) = [SP++];   // Restore registers from stack
        (R7:0) = [SP++];
        RETS = [SP++];
        ASTAT = [SP++];

/*********************************************************/
        RTS;
```