**Compile Using TI Optimizing Compiler for C6722 DSP 4-9-14**

Simple Benchmark loaded into Thread #2 – file = <Install>\C Programs\TI\_Compiler\BlinkFast.c

#include "KMotionDef.h"

// Benchmark 4 million loops with double precision math

main()

{

int i;

double k=0;

volatile double n;

for(;;)

{

SetBit(47);

SetBit(46);

for (i=0;i<2000000;i++) k+=i;

n+=k;

ClearBit(47);

ClearBit(46);

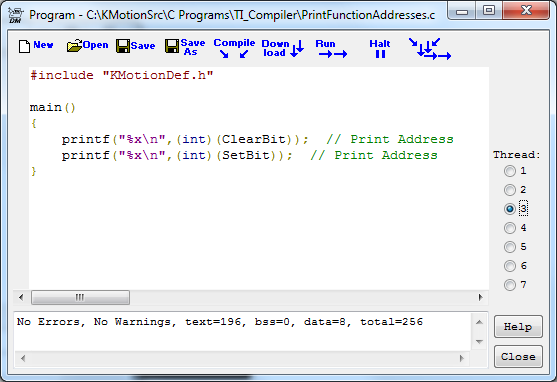
for (i=0;i<2000000;i++) k+=i;;

n+=k;

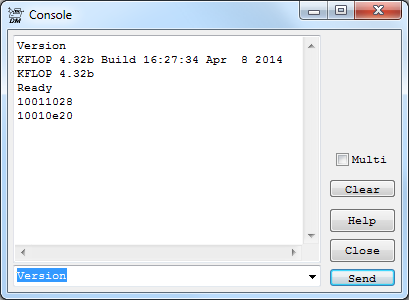
}

}

Determine addresses of any routines required in your program. In this case we need SetBit and ClearBit to Blink the LED. Use a simple program to determine the routine Addresses in KFLOP for the Version you have:



Record the Addresses from the Console:



Define the function symbols and their values as shown below in the TI Linker Command file: LnkThread2.cmd. This links the Code into some remaining unused Internal DSP RAM. The 256Kbytes of high speed Internal DSP RAM resides in address range 0x10000000 – 0x1001ffff

-c

-heap 15700000

-stack 0x800

/\*-lrts6701.lib \*/

/\* LINK CMD FILE TO PUT USER PROGRAM IN SMALL SPACE AT END OF INTERNAL DSP RAM !!!!!!!!!!!!!!!!! \*/

**/\* Hard Code KFLOP Addresses \*/**

**\_SetBit=0x10011028;**

**\_ClearBit=0x10010e20;**

MEMORY

{

IRAM\_BOOT: o = 10000000h l = 00001000h

ENTRYPT: o = 10001000h l = 00000040h

/\* IRAM: o = 10001040h l = 0001efc0h \*/

IRAM: o = 1001c000h l = 00004000h /\* for FAST User C Programs use small leftover toward end of IRAM !!!!!!!!!!!!\*/

The Batch file **MakeThread2.bat** can be used to Compile and Link the file BlinkFast.c into the binary file BlinkFast(2).out. Note the –o3 selects high optimization.

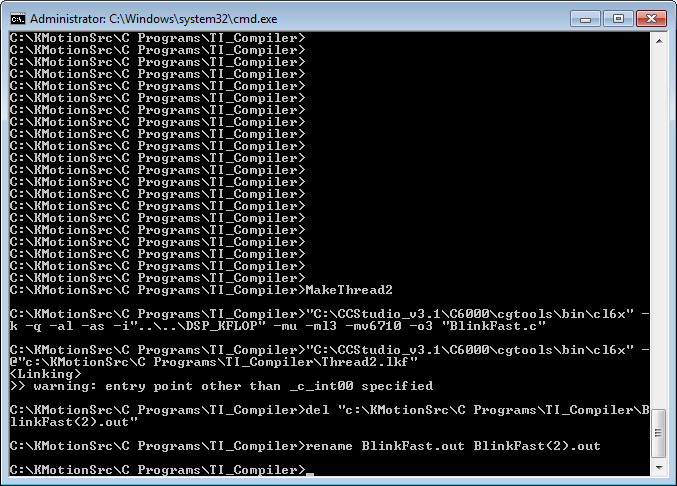
"C:\CCStudio\_v3.1\C6000\cgtools\bin\cl6x" -k -q -al -as -i"..\..\DSP\_KFLOP" -mu -ml3 -mv6710 -o3 "BlinkFast.c"

"C:\CCStudio\_v3.1\C6000\cgtools\bin\cl6x" -@"c:\KMotionSrc\C Programs\TI\_Compiler\Thread2.lkf"

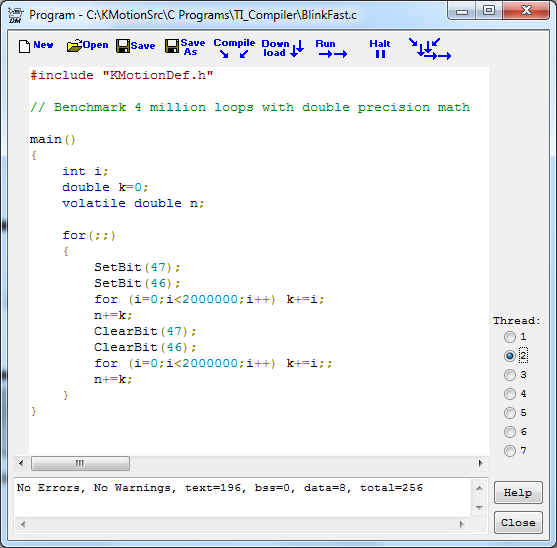
del "c:\KMotionSrc\C Programs\TI\_Compiler\BlinkFast(2).out"

rename BlinkFast.out BlinkFast(2).out

Ignore the warning regarding the C startup vector of \_c\_int00



Because we created the executable code with the same name BlinkFast(2).out as the standard compiler would create for Thread#2 we can use the Download and Run Buttons to execute the code. Note: do not push compile (or save/compile/download/run) or the TI generated binary will be overwritten by the standard TCC67 binary)



This code runs 24X faster. 24,000,000 loops per second while using only a fraction of the DSP’s time.

Each loop consists of:

* 32-bit integer count
* Integer to 64-bit double precision conversion
* 64-bit double addition
* Test
* Branch.