Benchmarking the Am186EM

Using the Dhrystone V2.1 as an example

Don Gille
Sr. Field Application Engineer
Advanced Micro Devices Inc.

Table of Contents

Overview

This application note is intended to demonstrate the processing power of the Am186EM microcontroller. By using a well-known and comparable benchmark, the Dhrystone, the evaluator can quickly determine the capability of the processor. The Dhrystone V2.1 benchmark may be replaced by another program more like the intended application or ... the application itself.

The evaluation kit is discussed first. Here the major functions and features are reviewed. A small amount of work is required to make the development platform ready to provide accurate timing measurements.

For those who wish to get a Quick Start, instructions are given to get the development board running pre-compiled Dhrystone in just a few minutes.

The tool set consists of a C++ compiler from Borland and a linker/locator from Paradigm. These state-of-the-art software systems provide the underlayment for the development of benchmarks and embedded programs.

An overview is offered of the timer/counter channels on the Am186EM. Covered are the capabilities, architecture and programming of each channel.

To facilitate the development process, the SD186EM board is programmed with PDREM. This FLASH based software allows source level debugging, easy loading and execution of benchmarks. The programmer is shown how to program PDREM into the FLASH memory on the SD186EM board.

After successfully running the Dhrystone, the programmer is guided to the point where he/she may run their own program. The step-by-step process of using the files developed for the Dhrystone on another program is explained.

Supplied files are covered as to their use and origin. This section also explains which files are expected from other sources.

Other information sources are covered in the last section.

Appendix A) The Dhrystone Benchmark is completely documented to help explain the benchmarking procedure. From this example the programmer can extract the necessary information to use all the support files for the Dhrystone on his/her own program.

Appendix B) A Summary of Processors is included to complete the Dhrystone picture. This listing contains various processors and their performance on the Dhrystone benchmark. Compiler flags and options are listed after the performance ranking, as well as, the site or person running the benchmark.

The Evaluation Kit

The Development Board

The SD186EM development board is a full function platform for code development, architectural evaluation and benchmarking. The board includes an Am186EM processor (40Mhz), memory, an RS-232 communications port, eight LEDs and a standardized bus connection.

The eight LEDs are CR2 through CR10. CR2 is optionally used to signal program start and CR3 is used to indicate program stop. Any LED can be turned off or on, using the functions provided, to indicate other program conditions.

The Serial UART Port

The Am186EM has an internal UART which is used to communicate with the board. This port can be polled or interrupt driven at user selectable speeds from 300 to 115,200 baud.

As shipped, the EMON monitor software operates the serial port at 19,200 baud. When the board is programmed with PDREM the port operates at 115,200 baud.

The Memory System

The SD186EM is supplied with 128K X 16 of 70ns AMD Flash and 128K X 16 of 70ns SRAM. This memory configuration allows the SD186EM platform to run full speed, making it suitable for performance benchmarking.

The Am186EM has programmable wait-state registers as a standard feature. The programmer wishing to use the platform for the purpose of benchmarking can initialize these registers before invoking the code under test. Wait-state selection inserts predictable memory delays allowing performance evaluation of proposed targets with less expensive memories.

Optional User Wiring

The evaluation kit, as supplied, does not connect timers 1 and 0 together. For short benchmarks with run times less than 40 seconds, no connection is necessary. For much longer run times, solder a jumper between pin 6 and pin 8 on row ‘B’ of J3 (the larger connector site at the opposite end of the board from the RS-232 connector). This connects the timer 1 output to the timer 0 input.

EMON Monitor

As delivered, the SD186EM development board powers up running the ‘EMON’ monitor. ‘EMON’ provides both debugging capability and user interface. Because the user interface is transmitted from the target board, only a simple modem communication or terminal emulation program needs to be run on the host.

Quick Start

To get the SD186EM up and running in the shortest amount of time, do the following:

1) Install all the required tools. For more information see the chapter titled “The Tool Set”.

- Paradigm PDREM
- Paradigm Locate
- Paradigm PDRT186 debugger
- Borland C++ V4.5

2) Edit changes in the correct files as covered in the chapter “Installation of PDREM” then compile, link and locate the file.

3) Program the new PDREM created above into the AMD FLASH memory on the SD186EM board following the instructions found in the same chapter under the heading of “Downloading to EMON”.

4) Make sure the port your board is connected to is the same port you called out in the installation of PDRT186. If not edit PDRT186.INI file and correct the port entry.

5) Run the debugger as shown in the index under the heading “Running Paradigms Debugger”.

After following these instructions, you will have developed a working software remote server, successfully programmed it into AMD FLASH memory, downloaded the Dhrystone benchmark and run the Dhrystone with results printed on your screen.

You are now ready to develop your own benchmark for the fastest 186 processor on earth!

The Tool Set

Requirements

It is expected that you own the tool set:

- Borland C++ V4.5 (optionally Microsoft C V8.0 or Visual C++ V2.0)

- Paradigm Locate V5.0, PDRT186 V4.0, PDREM

If you are using older versions of these tools, you must replace each file from Paradigm and Borland with the equivalent from your version.

NOTE: Be sure to read all the supplied *readme* files from each

Paradigm PDREM Remote Server

The SD186EM FLASH memory should be programmed with Paradigm's remote debug server PDREM. This ROM based support code assumes complete control of the SD186EM target board which allows the PC based debugger to operate the target over a serial link. PDREM is controlled by the PDRT186 debugger running on the host computer. Actions like single step, stop, run and read memory are supported by PDREM through the source level debugger.

Paradigm 'PDRT186' Debugger

This debugger is based on the Borland Turbo Debugger with extensions for the Am186EM processor. This debugger provides an easily understood user interface with specialized windows which make all the Am186EM internal registers visible.

Paradigm Locate

This tool is also required for **real mode** code development. Functioning as a second linker/locator, Locate takes an ".EXE" file and resolves jump targets to produce an executable file for an embedded design.

To produce a program which will run on the SD186EM, the programmer must prepare a configuration file for use by the locator. This file tells the locator where to place each section (code/data) in memory in the embedded system.

Borland C++ 4.5

The Borland environment, under Windows, gives the programmer an environment to develop functional DOS/Windows programs and then easily move them to the Am186EM embedded environment. Borland compilers give very good compile time performance, as well as, good optimization of code. Borland's tool kit along with the Paradigm locator, allows development of more complex C++ programs.

This application recommends that the Borland tool set be run from the command line in DOS. This was done to allow 'make' files to be easily used.

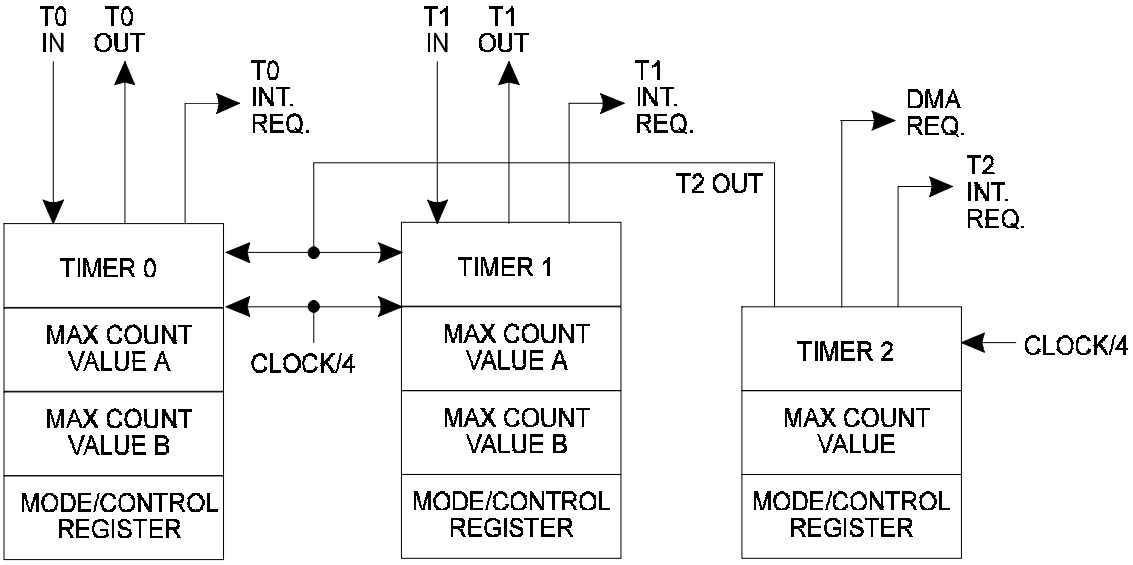
The 186EM Timing Unit

Capabilities

The Am186EM timer has exactly the same capabilities as the standard Am80186. These features include: 3 channels, 2 external pairs of inputs and outputs, programmable duty cycle and external or internal clocking. The Am186EM timers have also been enhanced to allow clocking at the extremely high speed the CPU is capable of, 40Mhz!

The timers will be used to measure the benchmarks run in this application note.

NOTE: If your benchmark run time is under 15 uS, place a loop around the code and run through it a number of times.



Addressing

The Am186EM uses a programmable address register to fix the upper 8 bits of the internal register array. It is assumed that this register is left in the default or reset state (FFh). The registers which control each timer are addressed by the address register as the upper 8 bits and the offset into the Am186EM's register array as the lower eight bits of the address.

The following 'C' code is an example which could be placed in a header file and included in any code using the timer unit.

The supplied programs use the header file "am186em.h" for all addressing.

// example header file for timer addressing

```
#define t0cnt 0ff50h // timer 0 count register
#define t1cnt 0ff58h // timer 1 count register
#define t2cnt 0ff60h // timer 2 count register
#define t0mode 0ff56h // timer 0 mode register
#define t1mode 0ff5eh // timer 1 mode register
#define t2mode 0ff66h // timer 2 mode register
#define t0maxa 0ff52h // timer 0 primary maximum count compare register
#define t1maxa 0ff5ah // timer 1 primary maximum count compare register
#define t2maxa 0ff62h // timer 2 primary maximum count compare register
#define t1maxb 0ff54h // timer 1 secondary maximum count compare register
#define t2maxb 0ff5ch // timer 2 secondary maximum count compare register
```

Programming

Operation of the timer unit is as simple as setting the correct registers. The code in this section will connect timer 2 to timer 1 which in-turn drives timer 0. The clock source for timer 2 is the pre-scaled (divide by 4) CPU clock or 10Mhz. Timer 2 divides this by 65535 and clocks timer 1. Timer 1 divides this by 65535 and clocks timer 0. This long chain allows benchmarks to run for a number of hours.

This code should be added to, and called by, your benchmark code. It was coded in assembly language to make sure it was as small and fast as possible. No call overhead is incurred because the compiler will inline the functions containing the assembly language. Also, compiler optimizations do not affect assembly coded routines.

The supplied programs use the 'C' language as the frame work while inserting assembly code into the program to do the actual work. Most 'C' compilers allow mixed language but their syntax may differ.

The timer operation code is in a file called timer.c and the header code is in a file called timer.h.

```
/*
        TIMER.H
        Timer counter function prototypes and structure type defs
*/

#ifndef TIMER_H
#define TIMER_H
```

```
// definitions
#define MAXCNT1A 0xFFFF //maximum count for register a of timer 1
#define MAXCNT1B 0xFFFF
#define MAXCNT0A 0xFFFF
#define MAXCNT0B 0xFFFF
#define MAXCNT2A 0xFFFF


// type definitions
typedef  struct {
                                        unsigned int t2; //fastest changing count
                                        unsigned int t1;
                                        unsigned int t0; //slowest count
                                        } T_Time;


// function prototypes
void timer_init(void);
void timer_start(void);
void timer_stop(void);
void read_time(T_Time *);
float elapsed_time(float cpu_clock,T_Time * end);


#endif //TIMER_H


/*
        Timer.c
        Timer functions for benchmarking
*/

#pragma inline
#include "asmrules.h"
#include "am186em.h"
#include "timer.h"

#define TIMER_C

/*
        initialize the timers
*/
void timer_init(void){
asm mov dx,T2MODE //point to mode register
asm mov ax,0100000000000001b //prescale mode CPU/4 as input
asm out dx,ax
asm mov dx,T1MODE //point to mode register
asm mov ax,1100000000001001b //t2 internal select as source (tie input pin low)
asm out dx,ax
```

```
asm mov dx,T0MODE //point to mode register
asm mov ax,1100000000000101b //t1 as input (wire t1 out to t0 in)
asm out dx,ax
// timer 2 stopped, timers 1 and 0 enabled waiting on t2 to start

// initialize count and max registers
// zero the current count registers
asm xor ax,ax // 0 for storage
asm mov dx,T2CNT
asm out dx,ax
asm mov dx,T1CNT
asm out dx,ax
asm mov dx,T0CNT
asm out dx,ax

// setup maximum count registers
asm mov ax,MAXCNT2A
asm mov dx,T2MAXA
asm out dx,ax
asm mov ax,MAXCNT1A
asm mov dx,T1MAXA
asm out dx,ax
asm mov ax,MAXCNT0A
asm mov dx,T0MAXA
asm out dx,ax
asm mov ax,MAXCNT1B
asm mov dx,T1MAXB
asm out dx,ax
asm mov ax,MAXCNT0B
asm mov dx,T0MAXB
asm out dx,ax

// setup PIOs for t1 -> t0

// set p11 to input (TMRIN0) and set p1 to output as TMROUT1
asm mov dx,PIO0EN
asm in ax,dx
asm mov bx,1111011111111101b
asm and ax,bx
asm out dx,ax
}

void timer_start(void){
asm mov dx,T2MODE
asm mov ax,1100000000000001b // start up counter 2
asm out dx,ax
}

void timer_stop(void){
asm mov dx,T2MODE
```

```
asm mov ax,0100000000000001b
asm out dx,ax
}


void read_time(T_Time * emtime){
asm LES_ di,emtime //get pointer
asm mov dx,T2CNT //point to count register
asm in ax,dx
asm mov ES_ [di],ax //first member
// use all three counters in a serial chain
asm mov dx,T1CNT
asm in ax,dx
asm mov ES_ [di+2],ax //second member
asm mov dx,T0CNT
asm in ax,dx
asm mov ES_ [di+4],ax //third member
#endif
}

#ifdef ET
/*
        elapsed_time
        calculate the time in seconds from a T-Type structure
        as a floating point number of seconds
        assumes timer initialized to 0 before start.
*/

float elapsed_time(float cpu_clock,T_Time * end){
        float t2clx,t1clx,t2time;

        float t0clx,t1time,t0time;
        t2clx = cpu_clock/4.0; // calculate clock to each timer stage
        t1clx = t2clx/(float)MAXCNT2A;
        t0clx = t1clx/(float)MAXCNT1A;
        t2time = (1/t2clx) * (float)end->t2; // in rev e timers chained inline
        t1time = (1/t1clx) * (float)end->t1;
        t0time = (1/t0clx) * (float)end->t0;
        return(t2time+t1time+t0time);
}
#endif
```

Dhrystone Source Code

```
/* Modified timer start/stop for AMD demo */
/*
 ***************************************************************************
 *
 *                  "DHRYSTONE" Benchmark Program
 *                  -----------------------------
 *
 *  Version:    C, Version 2.1
 *
 *  File:       dhry_1.c (part 2 of 3)
 *
 *  Date:       May 25, 1988
 *
 *  Author:     Reinhold P. Weicker
 *
 ***************************************************************************
 */



#include "dhry.h"

// AMD includes
#include "timer.h"
#include "eleds.h"



/* Global Variables: */

Rec_Pointer     Ptr_Glob, Next_Ptr_Glob;
int             Int_Glob;
Boolean         Bool_Glob;
char            Ch_1_Glob, Ch_2_Glob;
#if 0
int             Arr_1_Glob[50];
int             Arr_2_Glob[50][50];
#else
unsigned int    Arr_1_Glob[50];
unsigned int    Arr_2_Glob[50][50];
#endif
Boolean         Reg = DEF_REG;

/* variables for time measurement: */


/* end of variables for time measurement */


void main()
/*****/

 /* main program, corresponds to procedures        */
 /* Main and Proc_0 in the Ada version             */
```

```
{
        One_Fifty       Int_1_Loc;
        REG One_Fifty   Int_2_Loc;
        One_Fifty       Int_3_Loc;
        REG char        Ch_Index;
        Enumeration     Enum_Loc;
        Str_30          Str_1_Loc;
        Str_30          Str_2_Loc;
#if 0
        // REG int         Run_Index;
        // REG int         Number_Of_Runs;
#else
        // REG unsigned int Run_Index;
        // REG unsigned int Number_Of_Runs;
#endif

        long Number_Of_Runs,Run_Index; // added by AMD

#ifdef AM186EM
        // variables used
        T_Time Ttest_time;
        float loop_time,total_time,bench_time;
        long amd;
        //initialization
        ledinit(); //initialize outputs for AM186EM, AMD
        nowait(); // set wait states to 0 for SRAM
        timer_init(); //setup timer and zero
#endif


        /* Initializations */

        Next_Ptr_Glob = (Rec_Pointer) malloc(sizeof(Rec_Type));
        Ptr_Glob = (Rec_Pointer) malloc(sizeof(Rec_Type));

        Ptr_Glob->Ptr_Comp = Next_Ptr_Glob;
        Ptr_Glob->Discr = Ident_1;
        Ptr_Glob->variant.var_1.Enum_Comp = Ident_3;
        Ptr_Glob->variant.var_1.Int_Comp = 40;
        strcpy(Ptr_Glob->variant.var_1.Str_Comp,
          "DHRYSTONE PROGRAM, SOME STRING");
        strcpy(Str_1_Loc, "DHRYSTONE PROGRAM, 1'ST STRING");

        Arr_2_Glob[8][7] = 10;
        /* Was missing in published program. Without this statement,    */
        /* Arr_2_Glob [8][7] would have an undefined value.             */
        /* Warning: With 16-Bit processors and Number_Of_Runs > 32000,  */
        /* overflow may occur for this array element.                   */


        printf("\nDhrystone Benchmark, Version 2.1 (Language: C)\n");
        if (Reg) {
        printf("Program compiled with 'register' attribute\n");
        } else {
        printf("Program compiled without 'register' attribute\n");
        }
```

```
        Number_Of_Runs = 6000; //set to your liking


        printf("\nExecution starts, %u runs through Dhrystone\n", Number_Of_Runs);


        /***************/
        /* Start timer */
        /***************/
#ifdef AM186EM
        //test loop time
        amd = 0;
        timer_start();
        // could also use a pragma here to disable optimization
        for (Run_Index = 1; Run_Index <= Number_Of_Runs; ++Run_Index){
                amd = amd +1; // keeps optimizer at bay
        }
        timer_stop();
        amd = amd+2; // keeps optimizer at bay
        read_time(&Ttest_time); //get counters into loop_time struct
        loop_time = elapsed_time(40.0e6,&Ttest_time);
        timer_init(); //reset the timer
        //now start the program
        cr2_on(); // signal start of program
        timer_start(); //start time chain
#endif
        /***************/

        for (Run_Index = 1; Run_Index <= Number_Of_Runs; ++Run_Index) {

        Proc_5();
        Proc_4();
        /* Ch_1_Glob == 'A', Ch_2_Glob == 'B', Bool_Glob == true */
        Int_1_Loc = 2;
        Int_2_Loc = 3;
        strcpy(Str_2_Loc, "DHRYSTONE PROGRAM, 2'ND STRING");
        Enum_Loc = Ident_2;
        Bool_Glob = !Func_2(Str_1_Loc, Str_2_Loc);
        /* Bool_Glob == 1 */
        while (Int_1_Loc < Int_2_Loc) { /* loop body executed once */
                Int_3_Loc = 5 * Int_1_Loc - Int_2_Loc;
                /* Int_3_Loc == 7 */
                Proc_7(Int_1_Loc, Int_2_Loc, &Int_3_Loc);
                /* Int_3_Loc == 7 */
                Int_1_Loc += 1;
        }       /* while */
        /* Int_1_Loc == 3, Int_2_Loc == 3, Int_3_Loc == 7 */
        Proc_8(Arr_1_Glob, Arr_2_Glob, Int_1_Loc, Int_3_Loc);
        /* Int_Glob == 5 */
        Proc_1(Ptr_Glob);
        for (Ch_Index = 'A'; Ch_Index <= Ch_2_Glob; ++Ch_Index)
                /* loop body executed twice */
        {
```

```
                if (Enum_Loc == Func_1(Ch_Index, 'C'))
                /* then, not executed */
                {
                 Proc_6(Ident_1, &Enum_Loc);
                strcpy(Str_2_Loc, "DHRYSTONE PROGRAM, 3'RD STRING");
                Int_2_Loc = Run_Index;
                Int_Glob = Run_Index;
                }
        }
        /* Int_1_Loc == 3, Int_2_Loc == 3, Int_3_Loc == 7 */
        Int_2_Loc = Int_2_Loc * Int_1_Loc;
        Int_1_Loc = Int_2_Loc / Int_3_Loc;
        Int_2_Loc = 7 * (Int_2_Loc - Int_3_Loc) - Int_1_Loc;
        /* Int_1_Loc == 1, Int_2_Loc == 13, Int_3_Loc == 7 */
        Proc_2(&Int_1_Loc);
        /* Int_1_Loc == 5 */

        }             /* loop "for Run_Index" */

        /**************/
        /* Stop timer */
        /**************/
#ifdef AM186EM
        timer_stop(); //stop timer
        cr3_on(); // signal program stop
        read_time(&Ttest_time);
        total_time = elapsed_time(40.0e6,&Ttest_time);
#endif

        /**************/

        /*****************/
        /* Calculations */
        /*****************/
#ifdef AM186EM
        bench_time = total_time-loop_time;
#endif
        /*****************/


        printf("Execution ends\n\n");

#ifdef DETAILS // must be defined to print all this stuff
        printf("Final values of the variables used in the benchmark:\n\n");
        printf("Int_Glob:            %d\n", Int_Glob);
        printf("        should be:   %d\n", 5);
        printf("Bool_Glob:           %d\n", Bool_Glob);
        printf("        should be:   %d\n", 1);
        printf("Ch_1_Glob:           %c\n", Ch_1_Glob);
        printf("        should be:   %c\n", 'A');
        printf("Ch_2_Glob:           %c\n", Ch_2_Glob);
        printf("        should be:   %c\n", 'B');
        printf("Arr_1_Glob[8]:       %d\n", Arr_1_Glob[8]);
        printf("        should be:   %d\n", 7);
        printf("Arr_2_Glob[8][7]:    %d\n", Arr_2_Glob[8][7]);
```

```
        printf("       should be:   Number_Of_Runs + 10\n");
        printf("Ptr_Glob->\n");
        printf("  Ptr_Comp:          %d\n", (int) Ptr_Glob->Ptr_Comp);
        printf("       should be:   (implementation-dependent)\n");
        printf("  Discr:            %d\n", Ptr_Glob->Discr);
        printf("       should be:   %d\n", 0);
        printf("  Enum_Comp:         %d\n", Ptr_Glob->variant.var_1.Enum_Comp);
        printf("       should be:   %d\n", 2);
        printf("  Int_Comp:          %d\n", Ptr_Glob->variant.var_1.Int_Comp);
        printf("       should be:   %d\n", 17);
        printf("  Str_Comp:          %s\n", Ptr_Glob->variant.var_1.Str_Comp);
        printf("       should be:   DHRYSTONE PROGRAM, SOME STRING\n");
        printf("Next_Ptr_Glob->\n");
        printf("  Ptr_Comp:          %d\n", (int) Next_Ptr_Glob->Ptr_Comp);
        printf("       should be:   (implementation-dependent), same as above\n");
        printf("  Discr:            %d\n", Next_Ptr_Glob->Discr);
        printf("       should be:   %d\n", 0);
        printf("  Enum_Comp:         %d\n", Next_Ptr_Glob->variant.var_1.Enum_Comp);
        printf("       should be:   %d\n", 1);
        printf("  Int_Comp:          %d\n", Next_Ptr_Glob->variant.var_1.Int_Comp);
        printf("       should be:   %d\n", 18);
        printf("  Str_Comp:          %s\n",
                Next_Ptr_Glob->variant.var_1.Str_Comp);
        printf("       should be:   DHRYSTONE PROGRAM, SOME STRING\n");
        printf("Int_1_Loc:           %d\n", Int_1_Loc);
        printf("       should be:   %d\n", 5);
        printf("Int_2_Loc:           %d\n", Int_2_Loc);
        printf("       should be:   %d\n", 13);
        printf("Int_3_Loc:           %d\n", Int_3_Loc);
        printf("       should be:   %d\n", 7);
        printf("Enum_Loc:            %d\n", Enum_Loc);
        printf("       should be:   %d\n", 1);
        printf("Str_1_Loc:           %s\n", Str_1_Loc);
        printf("       should be:   DHRYSTONE PROGRAM, 1'ST STRING\n");
        printf("Str_2_Loc:           %s\n", Str_2_Loc);
        printf("       should be:   DHRYSTONE PROGRAM, 2'ND STRING\n\n");
#endif



        printf("Dhrystone time = %f\n",bench_time);
        printf("Dhrystones / Second = %8.0f\n",bench_time/(float)Number_Of_Runs);
        printf("Dhrystone MIPS = %4.1f\n",((float)Number_Of_Runs/bench_time)/1757.0);

        exit(0);
}


void Proc_1(Ptr_Val_Par)
/******************/

        REG Rec_Pointer Ptr_Val_Par;
 /* executed once */
{
        REG Rec_Pointer Next_Record = Ptr_Val_Par->Ptr_Comp;
        /* == Ptr_Glob_Next */
```

```
        /* Local variable, initialized with Ptr_Val_Par->Ptr_Comp,    */
        /* corresponds to "rename" in Ada, "with" in Pascal           */

        structassign(*Ptr_Val_Par->Ptr_Comp, *Ptr_Glob);
        Ptr_Val_Par->variant.var_1.Int_Comp = 5;
        Next_Record->variant.var_1.Int_Comp
        = Ptr_Val_Par->variant.var_1.Int_Comp;
        Next_Record->Ptr_Comp = Ptr_Val_Par->Ptr_Comp;
        Proc_3(&Next_Record->Ptr_Comp);
        /*
         * Ptr_Val_Par->Ptr_Comp->Ptr_Comp == Ptr_Glob->Ptr_Comp
         */
        if (Next_Record->Discr == Ident_1)
        /* then, executed */
        {
        Next_Record->variant.var_1.Int_Comp = 6;
        Proc_6(Ptr_Val_Par->variant.var_1.Enum_Comp,
                    &Next_Record->variant.var_1.Enum_Comp);
        Next_Record->Ptr_Comp = Ptr_Glob->Ptr_Comp;
        Proc_7(Next_Record->variant.var_1.Int_Comp, 10,
                    &Next_Record->variant.var_1.Int_Comp);
        } else          /* not executed */
        structassign(*Ptr_Val_Par, *Ptr_Val_Par->Ptr_Comp);
}           /* Proc_1 */


void Proc_2(Int_Par_Ref)
/******************/
 /* executed once */
 /* *Int_Par_Ref == 1, becomes 4 */

        One_Fifty      *Int_Par_Ref;
{
        One_Fifty       Int_Loc;
        Enumeration     Enum_Loc;

        Int_Loc = *Int_Par_Ref + 10;
        do              /* executed once */
        if (Ch_1_Glob == 'A')
                /* then, executed */
        {
                Int_Loc -= 1;
                *Int_Par_Ref = Int_Loc - Int_Glob;
                Enum_Loc = Ident_1;
        }          /* if */
        while (Enum_Loc != Ident_1);/* true */
}           /* Proc_2 */


void Proc_3(Ptr_Ref_Par)
/******************/
 /* executed once */
 /* Ptr_Ref_Par becomes Ptr_Glob */

        Rec_Pointer    *Ptr_Ref_Par;
```

```
{
        if (Ptr_Glob != Null)
        /* then, executed */
        *Ptr_Ref_Par = Ptr_Glob->Ptr_Comp;
        Proc_7(10, Int_Glob, &Ptr_Glob->variant.var_1.Int_Comp);
}       /* Proc_3 */


void Proc_4()
{       /* without parameters */
        /*******/
        /* executed once */
        Boolean         Bool_Loc;

        Bool_Loc = Ch_1_Glob == 'A';
        Bool_Glob = Bool_Loc | Bool_Glob;
        Ch_2_Glob = 'B';
}       /* Proc_4 */


void Proc_5()
{       /* without parameters */
        /*******/
        /* executed once */
        Ch_1_Glob = 'A';
        Bool_Glob = false;
}       /* Proc_5 */


 /* Procedure for the assignment of structures,          */
 /* if the C compiler doesn't support this feature       */
#ifdef  NOSTRUCTASSIGN
memcpy(d, s, l)
        register char  *d;
        register char  *s;
        register int    l;
{
        while (l--)
        *d++ = *s++;
}
#endif


/*
 ********************************************************************************
 *
 *                  "DHRYSTONE" Benchmark Program
 *                  -----------------------------
 *
 *  Version:    C, Version 2.1
 *
 *  File:       dhry_2.c (part 3 of 3)
 *
 *  Date:       May 25, 1988
 *
 *  Author:     Reinhold P. Weicker
```

```
 *
 ****************************************************************************
 */

#ifndef REG
#define REG
 /* REG becomes defined as empty */
 /* i.e. no register variables   */
#ifdef _AM29K
#undef REG
#define REG register    /* Define REG; saves room on 127-char MS-DOS cmd line */
#endif
#endif

#include "dhry.h"

extern int      Int_Glob;
extern char     Ch_1_Glob;


void Proc_6(Enum_Val_Par, Enum_Ref_Par)
/*********************************/
 /* executed once */
 /* Enum_Val_Par == Ident_3, Enum_Ref_Par becomes Ident_2 */

        Enumeration     Enum_Val_Par;
        Enumeration    *Enum_Ref_Par;
{
        *Enum_Ref_Par = Enum_Val_Par;
        if (!Func_3(Enum_Val_Par))
        /* then, not executed */
        *Enum_Ref_Par = Ident_4;
        switch (Enum_Val_Par) {
        case Ident_1:
        *Enum_Ref_Par = Ident_1;
        break;
        case Ident_2:
        if (Int_Glob > 100)
                /* then */
                *Enum_Ref_Par = Ident_1;
        else
                *Enum_Ref_Par = Ident_4;
        break;
        case Ident_3:       /* executed */
        *Enum_Ref_Par = Ident_2;
        break;
        case Ident_4:
        break;
        case Ident_5:
        *Enum_Ref_Par = Ident_3;
        break;
        }               /* switch */
}           /* Proc_6 */


void Proc_7(Int_1_Par_Val, Int_2_Par_Val, Int_Par_Ref)
```

```
/*************************************************/
 /* executed three times                                 */
 /* first call:      Int_1_Par_Val == 2, Int_2_Par_Val == 3,  */
 /* Int_Par_Ref becomes 7                 */
 /* second call:     Int_1_Par_Val == 10, Int_2_Par_Val == 5, */
 /* Int_Par_Ref becomes 17                */
 /* third call:      Int_1_Par_Val == 6, Int_2_Par_Val == 10, */
 /* Int_Par_Ref becomes 18                */
          One_Fifty       Int_1_Par_Val;
          One_Fifty       Int_2_Par_Val;
          One_Fifty      *Int_Par_Ref;
{
          One_Fifty       Int_Loc;

          Int_Loc = Int_1_Par_Val + 2;
          *Int_Par_Ref = Int_2_Par_Val + Int_Loc;
}            /* Proc_7 */


void Proc_8(Arr_1_Par_Ref, Arr_2_Par_Ref, Int_1_Par_Val, Int_2_Par_Val)
/*********************************************************************/
 /* executed once      */
 /* Int_Par_Val_1 == 3 */
 /* Int_Par_Val_2 == 7 */
          Arr_1_Dim       Arr_1_Par_Ref;
          Arr_2_Dim       Arr_2_Par_Ref;
          int             Int_1_Par_Val;
          int             Int_2_Par_Val;
{
          REG One_Fifty   Int_Index;
          REG One_Fifty   Int_Loc;

          Int_Loc = Int_1_Par_Val + 5;
          Arr_1_Par_Ref[Int_Loc] = Int_2_Par_Val;
          Arr_1_Par_Ref[Int_Loc + 1] = Arr_1_Par_Ref[Int_Loc];
          Arr_1_Par_Ref[Int_Loc + 30] = Int_Loc;
          for (Int_Index = Int_Loc; Int_Index <= Int_Loc + 1; ++Int_Index)
          Arr_2_Par_Ref[Int_Loc][Int_Index] = Int_Loc;
          Arr_2_Par_Ref[Int_Loc][Int_Loc - 1] += 1;
          Arr_2_Par_Ref[Int_Loc + 20][Int_Loc] = Arr_1_Par_Ref[Int_Loc];
          Int_Glob = 5;
}            /* Proc_8 */


Enumeration Func_1(Ch_1_Par_Val, Ch_2_Par_Val)
/*************************************************/
 /* executed three times                                 */
 /* first call:      Ch_1_Par_Val == 'H', Ch_2_Par_Val == 'R'    */
 /* second call:     Ch_1_Par_Val == 'A', Ch_2_Par_Val == 'C'    */
 /* third call:      Ch_1_Par_Val == 'B', Ch_2_Par_Val == 'C'    */

          Capital_Letter  Ch_1_Par_Val;
          Capital_Letter  Ch_2_Par_Val;
{
          Capital_Letter  Ch_1_Loc;
          Capital_Letter  Ch_2_Loc;
```

```
        Ch_1_Loc = Ch_1_Par_Val;
        Ch_2_Loc = Ch_1_Loc;
        if (Ch_2_Loc != Ch_2_Par_Val)
        /* then, executed */
        return (Ident_1);
        else {          /* not executed */
        Ch_1_Glob = Ch_1_Loc;
        return (Ident_2);
        }
}           /* Func_1 */


Boolean Func_2(Str_1_Par_Ref, Str_2_Par_Ref)
/*************************************************/
 /* executed once */
 /* Str_1_Par_Ref == "DHRYSTONE PROGRAM, 1'ST STRING" */
 /* Str_2_Par_Ref == "DHRYSTONE PROGRAM, 2'ND STRING" */

        Str_30          Str_1_Par_Ref;
        Str_30          Str_2_Par_Ref;
{
        REG One_Thirty  Int_Loc;
        Capital_Letter  Ch_Loc;

        Int_Loc = 2;
        while (Int_Loc <= 2)    /* loop body executed once */
        if (Func_1(Str_1_Par_Ref[Int_Loc],
                    Str_2_Par_Ref[Int_Loc + 1]) == Ident_1)
                /* then, executed */
        {
                Ch_Loc = 'A';
                Int_Loc += 1;
        }       /* if, while */
        if (Ch_Loc >= 'W' && Ch_Loc < 'Z')
        /* then, not executed */
        Int_Loc = 7;
        if (Ch_Loc == 'R')
        /* then, not executed */
        return (true);
        else {          /* executed */
        if (strcmp(Str_1_Par_Ref, Str_2_Par_Ref) > 0)
                /* then, not executed */
        {
                Int_Loc += 7;
                Int_Glob = Int_Loc;
                return (true);
        } else          /* executed */
                return (false);
        }           /* if Ch_Loc */
}           /* Func_2 */


Boolean Func_3(Enum_Par_Val)
/***************************/
 /* executed once        */
```

```
 /* Enum_Par_Val == Ident_3 */
        Enumeration     Enum_Par_Val;
{
        Enumeration     Enum_Loc;

        Enum_Loc = Enum_Par_Val;
        if (Enum_Loc == Ident_3)
        /* then, executed */
        return (true);
        else            /* not executed */
        return (false);
}           /* Func_3 */
```

Profile of Dhrystone

The Borland profiler tool tracks the percentage of time spent in each function of the program under test. Using Dhrystone as an example, the profilers output gives a good idea where time is spent in the benchmark. If you are looking for areas of improvement, the profiler tool can help you focus your efforts for maximum return.

To use the profiler, you must compile with Borland C++ under Windows so that it can be run under Windows. To do this select the option for *'Easy Win'* from the new project menu. This option allows you to run non-Windows programs under the Windows operating system.

Below is the output of the Borland C++ 4.5 profiler. The length of the bars (built from asterisks) determines the amount of CPU time consumed by the function.

Turbo Profiler for Windows Version 4.5 Tue Mar 28 19:44:07 1995

Program: C:\E86\PROG\DHRY21\DRY21.EXE

```
Execution Profile
Total time: 6.9416 sec
% of total: 98 %
       Run: 1 of 1

    Filter: All
      Show: Time
      Sort: Frequency

_main           3.1417 sec  45% |*******************************************
_Proc_1         1.4323 sec  20% |********************
_Proc_6         0.5511 sec   8% |********
_Func_2         0.4962 sec   7% |*******
_Proc_3         0.3859 sec   5% |*****
_Proc_7         0.2761 sec   4% |****
_Func_1         0.2759 sec   4% |****
_Proc_2         0.0553 sec  <1% |
_Proc_8         0.0552 sec  <1% |
_Func_3         0.0552 sec  <1% |
_Proc_5         0.0552 sec  <1% |
_Proc_4         0.0552 sec  <1% |
```

Installation of PDREM

Description

PDREM as it is called is the remote ROM based software which controls the operation of the SD186EM development board. Such actions as single step, break and dump registers are directly controlled by PDREM. This remote server communicates to the debugger running on the PC through the RS-232 port, 'COM 1' or 'COM 2'. Communication is carried out using binary packets which limits traffic on the serial cable making the debugger faster.

As supplied, the SD186EM board does not have PDREM installed. To use PDREM on the SD186EM requires that the software be compiled, linked/located then programmed into AMD Flash memory. The information below details the changes needed to accomplish this task.

Be sure to look over all the 'readme' files on the installation disk.

The PDREM Development Process

It is assumed that the Paradigm software was installed for the Am186EM.

1) Edit the configuration file as shown below

2) Edit the source files as shown below

3) Use "make" to re-compile, re-link and re-locate PDREM

4) Bring up 'EMON' using the Windows terminal package, or another modem communications package and download the HEX file you just made. More instructions are included in the SD186EM kit.

5) Cycle the power on your SD186EM board. After about a 15 second delay you should see the LEDs step one at a time toward the power connector.

You are now ready to run Paradigm Debug "PDRT186"

Configuration File

This configuration file directs Locate to place PDREM below EMON and to intercept the CPU as it jumps out of EMON. Please see the SD186EM kit documentation for details.

Please make sure your configuration matches the one provided, especially the bolded lines. This file is found in the "PDREM" directory built by the installation of the Paradigm tools.

File: pdrem.cfg

```
//
//    Paradigm LOCATE configuration file for building the PDREM kernel.
//    Copyright (C) 1994 Paradigm Systems.  All rights reserved.
//
//      This is an example configuration for the AMD Am186EM.
//
```

```
//      Please ensure that the example values shown are appropriate for
//      your target hardware.
//
//

hexfile  intel86

listfile segments publics  by address   // Create a segment map [publics]

cputype  Am186EM                // Select the processor type

initcode noreset                // Reset vector [reset]

                                                                        // Chip select
initializations EMON
initcode        umcs = 0xE004     \    // 128K ROM, 0 wait states
                        lmcs = 0x9FBC     \    // 128K RAM, 0 wait states
                        mpcs = 0x80BF     \
                        mmcs = 0x11F8     \
                        pacs = 0x007F


dup     DATA ROMDATA            // Make a copy of the initialized data

class   CODE = 0xf600           // Modify to set the EPROM address [fe00]
class   DATA = 0x0040           // And the RAM address
class   ??LOCATE = 0xf7ff       // Chip select initialization code [fff0]
// want class ??BOOT = 0xf7ff to startup with EMON Boot jump vector (new)

order   DATA            \     // Fix the class ordering of DGROUP
                        CONST           \
                        BSS             \
                        BSSEND          \
                        STACK

order   CODE            \     // Code, initialized data in EPROM
                        ROMDATA         \
                        ENDROMDATA

output  CODE            \     // Write to the output file
                        ??LOCATE        \
                        ROMDATA ENDROMDATA


// SD186EM below FLASH = 0xe0000
```

Source File Editing

To prepare your PDREM port for use on the SD186EM you must edit the following source files. These files are found in the PDREM directory built by the installation of the Paradigm tools.

File: dcomms.c

Pay close attention to bolded information, editing it into your file if not already there.

You may set your baud rate as high as 115,200 baud. Also selection of clock speed needs to be done (the SD186EM runs at 40Mhz). The Am186EM UART will be run in interrupt mode.

```
/*
//                      Paradigm PDREMOTE/ROM and TDREM Remote Target System Interface
//
//                      AMD Am186EM Serial Driver (polled/interrupt mode)
//
//                      Copyright (C) 1994 Paradigm Systems.  All rights reserved.
//
//                      This module provides support for the internal serial port on the
//                      AMD Am186EM microprocessor.  It can be operated in either polled
//                      or interrupt mode (by defining the symbol COMMINT in TARGET.H).
//
//   P1 does not require NULL Modem cable.
//
//                      Select desired BAUD Rate and Crystal Frequency below.
//
//
//
//                      Functions possibly requiring customization are
//                              COMMinit                        Hardware initialization (SIO/PIC)
//                              COMMsendchar            Write a character
//                              COMMrcvchar                     Read a character
//                              COMMrcvint                      Receive interrupt handler
//                              COMMgotpacket           Tests if a complete packet has been received
*/


/* Select Xtal frequency and desired BAUD Rate                  */
#define   CLK                   40000000UL              /* X1/X2 Crystal input frequency */
#define   BAUD                  115200UL                        /* Desired BAUD rate          */

#define IOB186EM        0xff00                  /* Location of PCB */

#include   <dos.h>                                                      /* Enable/disable functions */

#include "typedefs.h"                                           /* Common definitions and prototypes  */
#include   "target.h"                                           /* Target system specific information */
#include   "helpers.h"                                                  /* PDREMOTE/ROM helper functions
*/

#include   "Am186EM.h"                                                  /* Am186EM definitions */
```

…

File: target.h

Edit bold changes into your file.

```
/*
//      Paradigm Systems PDREMOTE/ROM Interface
//      Copyright (C) 1993, 1994 Paradigm Systems. All rights reserved.
//
//      This file defines the target system dependent portions of the
//      Paradigm Paradigm DEBUG Remote Interface.
*/

#if !defined(_TARGET)
#define _TARGET

#if 0                           /* Disabled: include if PDREMOTE/ROM UI is used */
#include    "kernel.h"              /* Application interface extensions */
```

```
#endif

#if 1                           /*0 = no, 1=yes Define if interrupt-driven serial I/O */
#define COMMINT     0x14            /* Change to desired interrupt vector */
#endif

...
```

You must make this change to enable printing to your screen in the debugger!

File: custom.mac

Bolded paths should point to your tools.

```
#
#  Macros customized for use with Borland C++ 4.5 and TASM
#  These must be customized to match your compiler/assembler/linker
#

ASM     =  tasm
CC      =  bcc
LINK    =  tlink
LIBS    =
INCLUDE  =  c:\bc45\include

MODEL   =  s                    #  Please do not change this!!!

CFLAGS   =  -c -m$(MODEL) -I$(INCLUDE) -Z -Od -f- -w -DBCPP40 -1-
AFLAGS   =  /mx
LFLAGS   =
LTERM    =
...
```

File: pdrem.cfg

Edit this file to match bolded data.

```
//
//          Paradigm LOCATE configuration file for building the PDREMOTE/ROM kernel.
//          Copyright (C) 1993 Paradigm Systems.  All rights reserved.
//

hexfile intel86                                                         // Intel hex output
listfile    segments            // Create a segment map

map    0x00000 to 0x003ff as reserved  // Interrupt vector table
map    0x00400 to 0x00fff as rdwr      // PDREMOTE/ROM data area
map    0x01000 to 0xdffff as reserved  // Reserved for applications
map    0xe0000 to 0xfffff as rdonly    // PDREMOTE/ROM kernel EPROM

cputype AM186EM                 // Select the processor type

initcode inbyte 0x1000          // inbyte is here to create ??LOCATE class

/* Chip select inititalization not necessary (flash utility sets up
//          chip selects).  However, there is a memory limitation on how many
// initcode parameters can be used at address 0xf7ff0 (will run into
//          flash downloader).  Since the flash downloader will jump to address
// 0xf7ff0, the initcode is used here to serve as a jump to pdremote's
// code.  The class, ??LOCATE, will contain the jump automatically.
//
//                  umcs = 0xfe38       \
//                  lmcs = 0x0ff8       \
//                  mpcs = 0x80bf               \
//                  pacs = 0x003f       \
*/
```

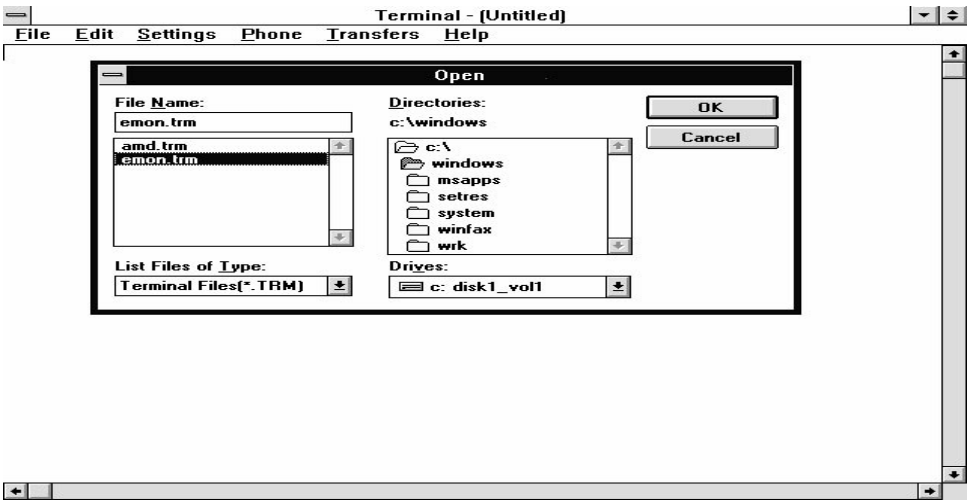
These are commented out, EMON has already set these up before jumping to PDREM.

```
dup         DATA ROMDATA          // Make a copy of the initialized data

class       CODE = 0xe000         // Modify to set the EPROM address
class       DATA = 0x0040         // And the RAM address
class       ??LOCATE = 0xf7ff     // Initialization code

order       DATA          \       // Fix the class ordering of DGROUP
                                              CONST         \
                                              BSS BSSEND    \
                                              STACK

order       CODE          \       // Code, initialized data in EPROM
                                              ROMDATA ENDROMDATA

output      CODE          \       // Write to the output file
                                              ??LOCATE      \
                                              ROMDATA ENDROMDATA
```

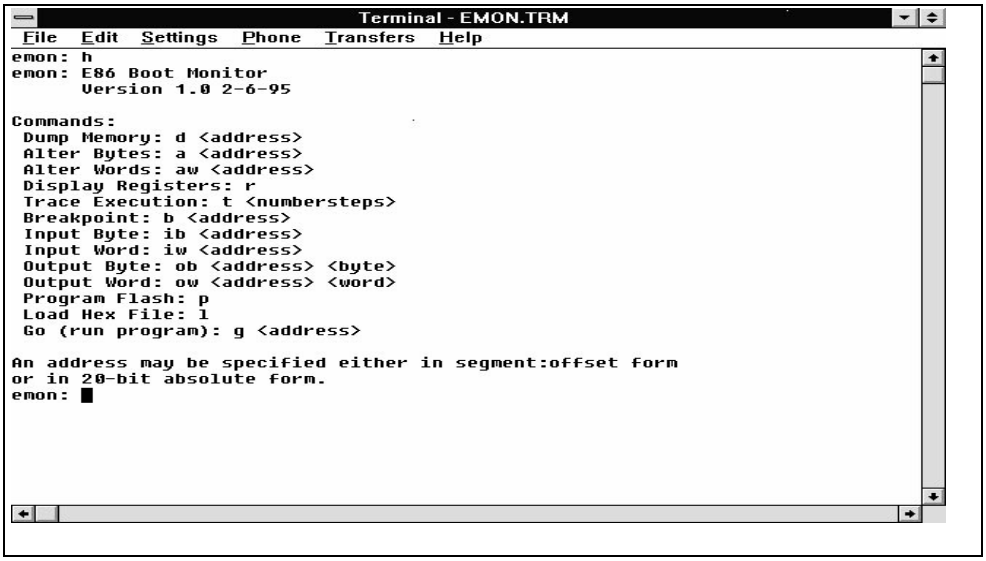
Downloading to EMON

The SD186EM development board is supplied with documentation on EMON. In this information you will find instructions on sending the PDREM.HEX file to the target. It is recommended that you use the Windows terminal package as the communications program. Supplied on the disk or the BBS files is a file emon.trm. This file configures the Windows terminal package to communicate with EMON and download the PDREM hex file.

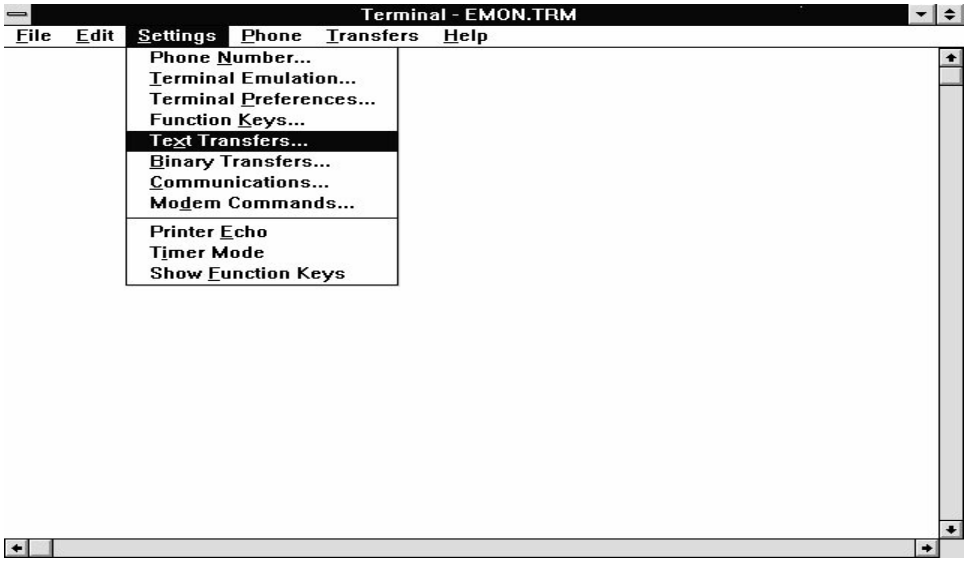
Click on the Windows terminal icon and the terminal emulator should open on your screen. Click on 'open' and select "emon.trm". The target board should be plugged into 'COM2'.



Cycle the power on the board, the LEDs should bounce back and forth once, press ‘cntl-break’ on your keyboard to signal EMON to begin operation. Once communication is established, the “emon:” prompt will be on your screen. If you type ‘h’ for help the following screen should appear.

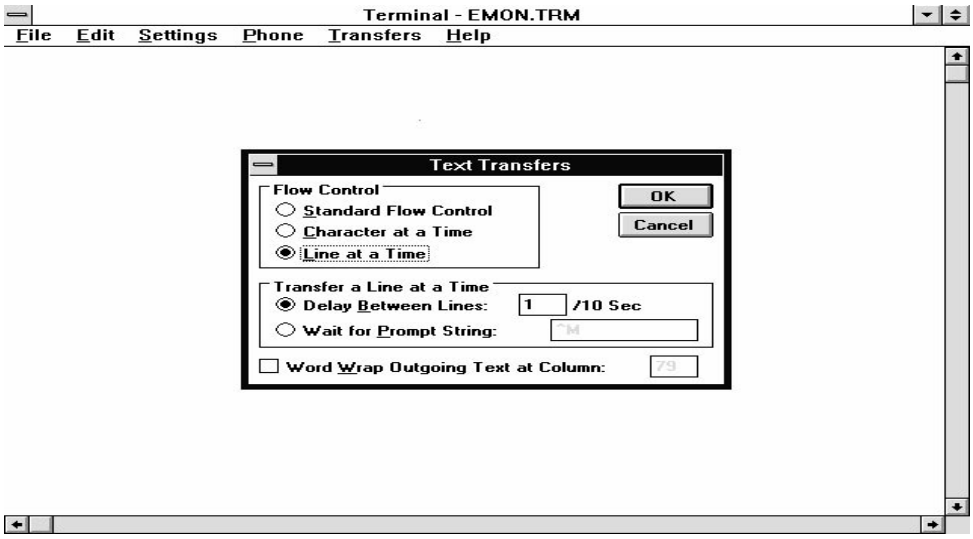


You must set up the Windows terminal package to transfer the HEX file to the target board. Click on ‘Settings’ then on ‘Text Transfers’ to set up the handshaking protocol.

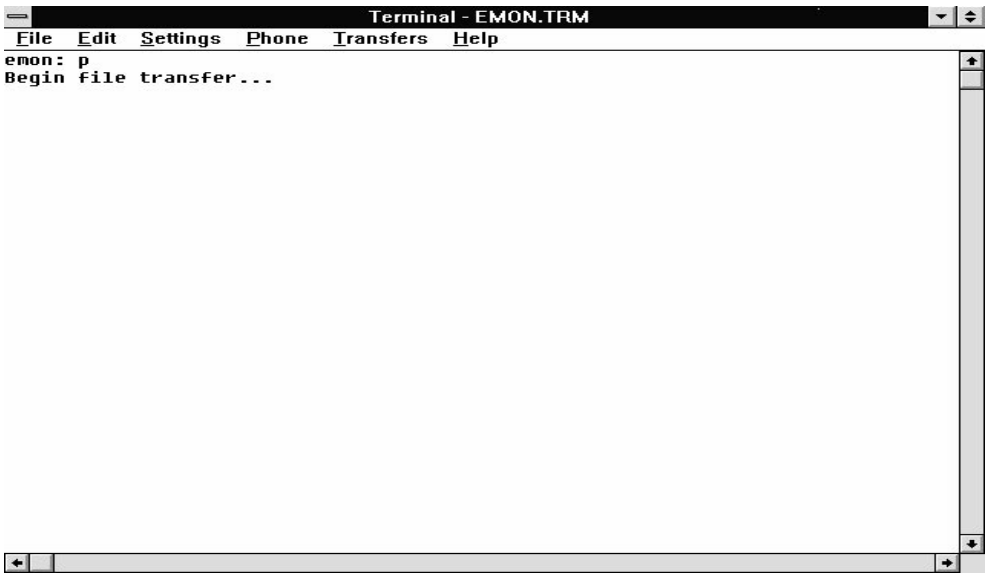


Setting a delay allows the Am186EM to write into memory with a comfortable margin of time to be ready for the next character sent.

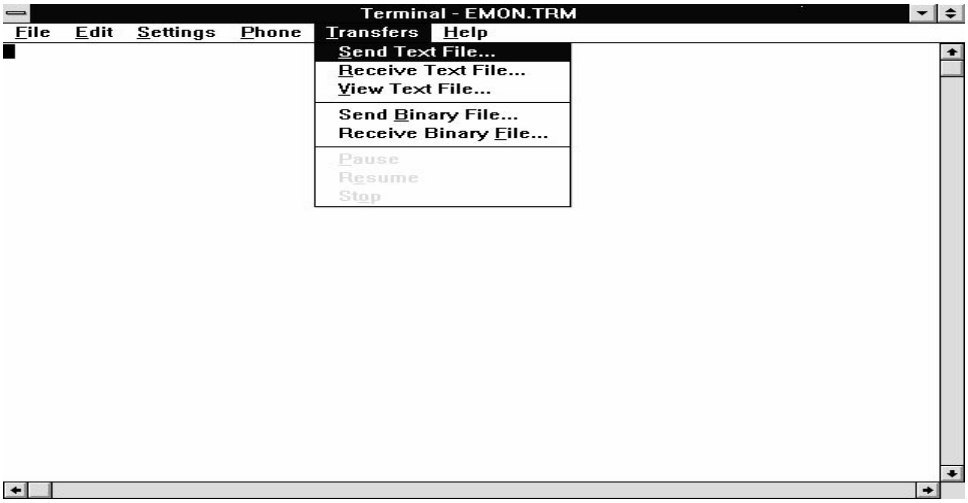
Set your Text Transfers box to look like this:



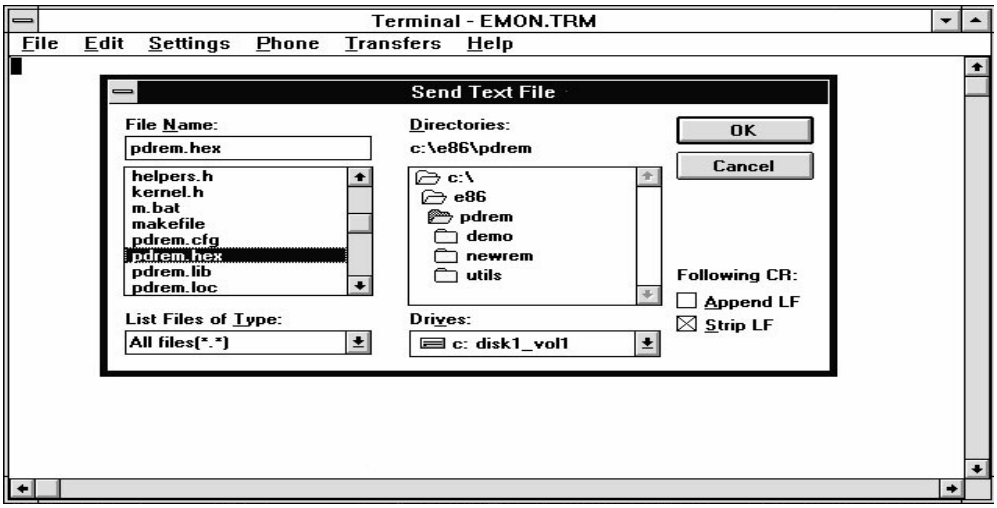
Once the terminal package is set up, type a ‘p’ command at the EMON prompt. This tells the monitor to receive a HEX file and program that file into FLASH memory. Once you hit the enter key your screen should look like this.



Now that ‘EMON’ is ready, transmit the HEX file to the target using the Windows terminal emulator. To do this click on ‘Transfers’ and in the pull down menu select ‘Send Text File’ as shown below.



Next you will be asked for a file name. To send the PDREMOTE.HEX file click in the directories box until your in the ‘PDREM’ directory then select ‘pdrem.hex’ from the file name list.



The file is first sent to the target, then EMON will print on your screen that the file was received and that programming had started. You will see the LEDs flashing randomly as the programming operation starts. When the LED activity stops, programming should be complete. EMON will indicate that the programming process is complete on your screen.

Running a Benchmark

The Benchmark Development Process

Follow these step to prepare your program for benchmarking:

1) Copy the AMD supplied files to your working directory.

- am186em.h	- asmrules.h
- efast.c	- support.h
-timer.c	- timer.h
- eleds.c	- makefile
- dhry21.cfg	- dhry21.mkf
- dhry21.rt	- dhry.h
- dhry_1.c	- dhry_2.c
- eleds.h	- paradigm.mkf

2) Copy these files from the "locate\bcpp45\helpers"(1), “pdrem\demo\console”(2) and "locate\bcpp45\examples\fpdemo"(3) directory to your working directory:

- bcpp45.asm (1)	- bcppsio.c (2)
- bcpp45.inc (1)	- console.c (2)
- bcppdmm.c (1)	- console.h (2)
- bcppflt.asm (3)	- dosemu.c (3)
- bcppflt.inc (3)	- dosemu.h (3)
-bcpprtl.asm (3)	- fardata.asm (3)
- startup.inc (3)	- fardata.cfg (3)
- typedefs.h (1)	- v8250.h (2)

3) Instrument the your main program.

4) Rename dhry21.rt to <your program name>.rt and edit the file as shown.

5) Edit the file named "makefile" as shown.

6) Edit "paradigm.mkf" as shown.

7) To invoke the tools simply type "make" at the DOS command line “c:\make”.

8) Load and run the code using Paradigm PDRT186.

As a final note, use of file I/O and the timers themselves by the benchmark is not allowed. The current development tools do not support disk file transactions such as 'fopen()' or 'fgetc()'. The timers are off-limits because they are being used to time the benchmark.

Instrumenting the Code

As can be seen in the "dhry_1.c " source code, all that is needed is to follow this simple template.

The AMD supplied files include a directory called 'bench'. In this directory is a set of files which allow you to simply edit your benchmark source code into an existing file "bench.c" (edit out what is already there). After you copy the needed Paradigm and Borland files into this directory you should be able to type 'make' at the DOS prompt to have a completed benchmark named "bench.axe".

In your include section:

#include "timer.h" // required for operation of the counter timers

#include "eleds.h" // optional, allows you to see start and stop of your program

In your main() section:

T_Time Ttest_time; // structure to hold the contents of the counter timer registers

float loop_time,total_time,bench_time; // times stored in seconds

In your initialization code section (inside main()):

ledinit(); // initialize the LED outputs

nowait(); // set the Am186EM to **0** wait states

timer_init(); // initialize the counter timers and set all counts to 0

Just before you start into the timed code:

If your benchmark requires multiple passes, you must insert this code. The time consumed in the loop overhead must be subtracted out of the total running time (as done here). If you do not have an outer loop like this omit this section of the code.

```
        /***************/
        /* Start timer */
        /***************/
#ifdef AM186EM // optional,   conditional compilation,
```

```
        //test loop time
// could use a #pragma here to turn off the optimizer
        amd = 0; // any variable will do here (unused one please)
        timer_start(); // init the timers and set to 0
// now do as many loops as the timed program will run.  If only 1 pass skip this
// section of code.
        for (Run_Index = 1; Run_Index <= Number_Of_Runs; ++Run_Index){
                amd = amd +1; // keeps optimizer at bay, it will try to take out code
        }
        timer_stop(); // stop the timer
        amd = amd+2; // keeps optimizer at bay
        read_time(&Ttest_time); //get counters into Ttest_time struct
        loop_time = elapsed_time(40.0,&Ttest_time); // produce a floating point
                                                    // number of seconds of execution
        timer_init(); //reset the timer back to 0
        //now start the program
        cr2_on(); // signal start of program turn on the LED
        timer_start(); //start timer again
#endif // optional, use if you used #ifdef
        /***************/
```

********** YOUR PROGRAM **********

<u>At the end of the program being timed:</u>

```
        /**************/
        /* Stop timer */
        /**************/
#ifdef AM186EM
        timer_stop(); //stop timer
        cr3_on(); // signal program stop
        read_time(&Ttest_time); // read the contents of the timers into a structure
        total_time = elapsed_time(40.0,&Ttest_time); // calculate number of secs.
#endif

        /**************/


        /****************/
        /* Calculations */
        /****************/
#ifdef AM186EM
```

```
// subtracts the loop overhead from the total time in the program.  Bench_time is
// the correct benchmark time.
        bench_time = total_time-loop_time;
#endif
        /***************/
```

Editing Configuration Files

File: dhry21.rt

Rename this file to <your_file_name>.rt and edit as shown

Configures paradigm locator for programs which are debugged in Paradigm Debug.

```
//
//          Paradigm LOCATE configuration file for debugging the Borland C++
//          application using Paradigm DEBUG/RT.
//

#include      "fardata.cfg"                         // Access the far data definitions

absfile axe86                               // Output for Paradigm DEBUG/RT
listfile      segments                          // Generate a segment map

map     0x01000 to 0x0ffff as rdwr                  // System RAM area (60KB RAM)
map     0x00000 to 0x00fff as reserved  // PDREMOTE and interrupt vector table
map     0x10000 to 0x3ffff as reserved  // No access allowed
map     0x40000 to 0x7ffff as rdonly    // Simulated EPROM area (256KB RAM)
map     0x80000 to 0xfffff as reserved  // No access allowed

cputype Am186EM

dup           DATA ROMDATA              // Make a copy of initialized data

class         CODE = 0x0500             //  loading at address
class         DATA = 0x0100               // Data  address

order         DATA                            \     // RAM class organization
              BSS BSSEND                      \
              STACK FARHEAP                   \
              _FARDATACLASSES

order         CODE                            \       // EPROM class organization
              INITDATA EXITDATA               \
              ROMDATA ENDROMDATA              \
              _ROMFARDATACLASSES

output  CODE                                  \     // Classes in the output file(s)
              INITDATA EXITDATA               \
              ROMDATA ENDROMDATA              \
              _ROMFARDATACLASSES
```

As your code and data grow these may need to be changed to reflect the new size of both the code and data sections of your program. Locate will tell you when they overlap.

Compilation

Compilation requires command line operation in DOS. (If you wish, the entire process can be done in the Windows IDE.) To invoke the process just type "make" on the DOS command line. Make will read a file in the current directory called "makefile". This file is used to prepare the benchmark for execution. Of note is the selection of optimization level (none). The optimizations called out on this line will have a great impact on performance and some times on code correctness. Always consult the compiler manual to understand the effect of the specified optimization. You may wish to see if code performance improves with other selections (recommended as an exercise).

File: makefile

```
#
#  Sample makefile for building the floating point arithmetic demonstration
#  using the Borland C++ compiler.
#
#  Some of the macros used in this makefile that can be customized are
#
#       MODEL           Selected memory model
#       CPU         CPU instruction selection
#       FLOAT           Floating point library selection
#       EXCEPTIONS  Exception handling selection
#       FARDATA     Enables use of class FAR_DATA, with optional compression
#       DEBUG           Enable/disable debug information
#       OPTIMIZE        Enable/disable optimization
#       WARNINGS        Enable/disable compiler warning messages
#

COMPDIR     =   c:\bc45     # Compiler home directory
BCCFG       =   turboc.cfg   # BC++ compiler configuration file
MKF         =   makefile   # Build everything if the makefile is changed

MODEL       =   s           # s, m, c, l, h
CPU         =   1           # 0 - 8086, 1 - 80186, 2 - 80286, 3 -80386
FLOAT       =   2           # 0 - none, 2 - emulator, 3 - coprocessor
OPTIMIZE    =   0           # 0 - none, 1 - size, 2 - speed
EXCEPTIONS  =   0           # 0 - disabled, 1 - enabled
FARDATA     =   1           # 0 - none, 1 - normal, 2 - compressed
                    # above must be 1 if float > 0

DEBUG       =   2           # 0 - none, 1 - debug EPROM, 2 - debug PDREMOTE
WARNINGS    =   1           # 0 - none, 1 - all

CODESTRING  =   0           # Put string literals in code segment
DUPSTRING   =   1           # Duplicate string merged
CHECKSTACK  =   0           # Check for stack overflow
STACK       =   8192        # Application stack size (in bytes)

!include    paradigm.mkf    # Compiler/linker customization options


#
#   These are the implicit rules for building C, C++, and assembly
#   language source modules.
#

.autodepend                 # Autodepedency checking
.suffixes:  .cpp .c .asm    # Rules when target's dependent is ambiguous
```

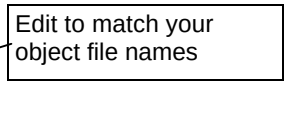
You must make sure that the paths point to your tool set directories!

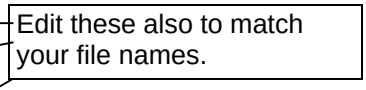
CPU must be 80186, to use the timer float must be set to 2, floating point requires fardata so set this to 1. Select your own optimization levels (per text).

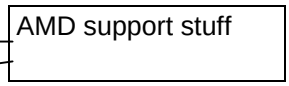
Select debug = 2 so you can use PDRT186

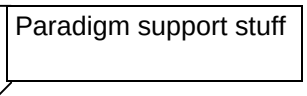
Watch your stack size!!!

```
.cpp.obj:
        bcc {$*.cpp }

.c.obj:
        bcc {$*.c }

.asm.obj:
        tasm $(AFLAGS) $*.asm


#
#   Here we have the list of object files that will form the
#   application.  STARTUP is special - it is used to make sure that
#   the startup code is linked first to set the order and alignment of
#   segments in the application.
#

STARTUP    =   bcpp45.obj fardata.obj
OBJS  = dosemu.obj dhry_1.obj dhry_2.obj console.obj bcppsio.obj bcpprtl.obj bcppflt.obj bcppdmm.obj efast.obj eleds.obj
timer.obj


#
#   The remainder of the make file is the targets and dependencies
#

dhry21.$(OUT):      dhry21.rom dhry21.$(CFG) fardata.cfg
        locate -cdhry21.$(CFG) $(PFLAGS) $*

dhry21.rom:         tlink.cfg $(STARTUP) $(FARDATAOBJ) $(OBJS)
        tlink @&&!
$(STARTUP) $(FARDATAOBJ) $(OBJS)
$*.rom
$*.map
$(LIBS)
!
dhry_1.obj:     dhry_1.c dhry.h $(MKF)
dhry_2.obj:     dhry_2.c dhry.h $(MKF)


timer.obj:      timer.c timer.h am186em.h $(MKF)
efast.obj:      efast.c am186em.h $(MKF)
eleds.obj:      eleds.c am186em.h $(MKF)

dosemu.obj:     dosemu.c typedefs.h dosemu.h $(BCCFG) $(MKF)

bcppflt.obj:    bcppflt.asm bcppflt.inc bcpp45.inc startup.inc $(MKF)

bcppsio.obj:    bcppsio.c bcpp45.inc startup.inc $(MKF)

bcpprtl.obj:    bcpprtl.asm bcpp45.inc startup.inc $(MKF)

fardata.obj:    fardata.asm bcpp45.inc startup.inc $(MKF)

bcpp45.obj:     bcpp45.asm bcpp45.inc startup.inc $(MKF)

console.obj:    console.c $(MKF)


#
#   Update the compiler/linker configuration files.  This ensures that a
#   default BC++ configuration file is never used to build an embedded
#   application.
#

turboc.cfg:         $(MKF)
  copy &&|
$(CFLAGS)
$(OPTFLAGS)
```









```
| turboc.cfg

tlink.cfg:        $(MKF)
  copy &&|
$(LFLAGS)
| tlink.cfg
```

File: paradigm.mkf

```
#
#   Borland C++ 4.5 Compiler/Linker command line option macros
#   Copyright (C) 1994 Paradigm Systems.  All rights reserved.
#

MKF        =   $(MKF) paradigm.mkf          # Include in dependancies

#
#   LIBS          List of libraries to be linked with the application
#   CFLAGS      Compiler command line options
#   AFLAGS      Assembler command line options
#   LFLAGS      Linker command line options
#   PFLAGS      Paradigm LOCATE command line options
#   OPTFLAGS        Compiler optimization options and other defines ....
#

LIBS          =   rc$(MODEL).lib

CFLAGS      =   -c -m$(MODEL) -I. -I$(COMPDIR)\include
AFLAGS      =   /mx /d__$(MODEL)__ /dSTKSIZE=$(STACK)
LFLAGS      =   /c /Lc:\bc45\lib
PFLAGS      =
OPTFLAGS       = -DAM186EM -DET

#
#   Process the debug options.  This step will optionally add debug
#   information to the compiler, assembler, and linker, plus select the
#   Paradigm LOCATE configuration file to build the application.
#

OUT         =   hex
CFG         =   rm
!if $(DEBUG) > 0
AFLAGS      =   $(AFLAGS) /zd
# CFLAGS      =   $(CFLAGS) -v -r-
CFLAGS      =   $(CFLAGS) -v
LFLAGS      =   $(LFLAGS) /v
!if $(DEBUG) == 2
AFLAGS      =   $(AFLAGS) /dPDREMOTE
CFLAGS      =   $(CFLAGS) -DPDREMOTE
OUT         =   axe
CFG         =   rt
!elif   $(DEBUG) > 2
!error  Invalid debug option selected
```

Keep these to enable the correct options

```
!endif
!endif

#
#   Process the selected CPU option
#

!if $(CPU) == 0
CFLAGS      =   $(CFLAGS) -1-
!elif   $(CPU) == 1
CFLAGS      =   $(CFLAGS) -1
!elif   $(CPU) == 2
CFLAGS      =   $(CFLAGS) -2
!elif   $(CPU) == 3
CFLAGS      =   $(CFLAGS) -3
!else
!error  Invalid CPU option selected
!endif

#
#   Process the selected exception handling option
#

!if $(EXCEPTIONS) == 0
CFLAGS      =   $(CFLAGS) -x-
LIBS        =   noeh$(MODEL).lib $(LIBS)
!elif   $(EXCEPTIONS) == 1
CFLAGS      =   $(CFLAGS) -x
AFLAGS      =   $(AFLAGS) /dEXCEPTIONS
!else
!error  Invalid exception option selected
!endif

#
#   Process the selected floating point option
#

!if $(FLOAT) == 0
CFLAGS      =   $(CFLAGS) -f-
!else
LIBS        =   remu.lib rmath$(MODEL).lib $(LIBS)
!if $(FLOAT) == 2
CFLAGS      =   $(CFLAGS) -f
AFLAGS      =   $(AFLAGS) /dFLOAT=$(FLOAT) /e
!elif   $(FLOAT) == 3
CFLAGS      =   $(CFLAGS) -f87
AFLAGS      =   $(AFLAGS) /dFLOAT=$(FLOAT) /r
!else
!error  Invalid floating point option selected
!endif
!endif

#
#   Process the FARDATA option.  If enabled, we include the module
#   FARDATA.OBJ in the object file list to copy or decompress class
#   FAR_DATA at run-time.
```

```
#

!if $(FARDATA) > 0
FARDATAOBJ  =   fardata.obj
PFLAGS      =   $(PFLAGS) -DHASFARDATA
!if $(FARDATA) == 2
AFLAGS      =   $(AFLAGS) /dCOMPRESSED
CFLAGS      =   $(CFLAGS) -DCOMPRESSED
PFLAGS      =   $(PFLAGS) -DCOMPRESSED
!elif   $(FARDATA) > 2
!error  Invalid FARDATA option selected
!endif
!endif

#
#   Process the advanced compiler options.
#

!if $(CODESTRING) == 1
CFLAGS      =   $(CFLAGS) -dc
!endif

!if $(DUPSTRING) == 1
CFLAGS      =   $(CFLAGS) -d
!endif

!if $(CHECKSTACK) == 1
CFLAGS      =   $(CFLAGS) -N
!endif

!if $(WARNINGS) == 0
CFLAGS  =   $(CFLAGS) -w-
!elif   $(WARNINGS) == 1
CFLAGS  =   $(CFLAGS) -w
!else
!error  Invalid WARNINGS option selected
!endif

!if $(OPTIMIZE) == 0
OPTFLAGS    =   $(OPTFLAGS) -Od
!elif   $(OPTIMIZE) == 1
OPTFLAGS    =   $(OPTFLAGS) -O1
!elif   $(OPTIMIZE) == 2
OPTFLAGS    =   $(OPTFLAGS) -O2
!else
!error  Invalid OPTIMIZE option selected
!endif
```

Carefully select these options or edit and change them only after you have read this document and your compiler manual carefully! (You could use -G for speed)

The following files are created by the “make” process, and are shown for completeness. You need not edit these files.

File: turboc.cfg

```
-c -ms -I. -Ic:\bc45\include -v -DPDREMOTE -1 -x- -f -d -w
```

-DAM186EM -DREV_E -DET -Od

File: tlink.cfg

/c /Lc:\bc45\lib /v

To make your own Dhrystone executable file just type the following at the DOS command line: C:\>make
You should see the following information on your screen as the 'make' process proceeds. If the order of compilation or assembly is different this does not indicate an error.

MAKE Version 3.7 Copyright (c) 1987, 1994 Borland International
tasm /mx /d__s__ /dSTKSIZE=8192 /zd /dPDREMOTE /dFLOAT=2 /e bcpp45.asm

Turbo Assembler Version 4.0 Copyright (c) 1988, 1993 Borland International

Assembling file: bcpp45.asm
Assembling for the small memory model
Paradigm LOCATE Borland C++ 4.5 Startup Support
Building Paradigm PDREMOTE-compatible startup code
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 222k

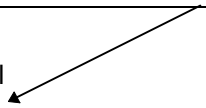
tasm /mx /d__s__ /dSTKSIZE=8192 /zd /dPDREMOTE /dFLOAT=2 /e fardata.asm

Turbo Assembler Version 4.0 Copyright (c) 1988, 1993 Borland International

Assembling file: fardata.asm
Assembling for the small memory model
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 228k

bcc dosemu.c dhry_1.c dhry_2.c console.c bcppsio.c

The warning messages below are present because the Dhrystone benchmark is not fully ANSI compliant. Dhrystone source does not include prototype for functions which results in these warnings.

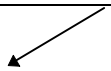


Borland C++ 4.5 Copyright (c) 1987, 1994 Borland International
dosemu.c:
dhry_1.c:
Warning dhry_1.c 81: Call to function 'nowait' with no prototype in function main
Warning dhry_1.c 88: Call to function 'malloc' with no prototype in function main
Warning dhry_1.c 89: Call to function 'malloc' with no prototype in function main
Warning dhry_1.c 96: Call to function 'strcpy' with no prototype in function main
Warning dhry_1.c 97: Call to function 'strcpy' with no prototype in function main
Warning dhry_1.c 113: Constant is long in function main
Warning dhry_1.c 144: Call to function 'Proc_5' with no prototype in function main
Warning dhry_1.c 145: Call to function 'Proc_4' with no prototype in function main
Warning dhry_1.c 149: Call to function 'strcpy' with no prototype in function main

```
Warning dhry_1.c 151: Call to function 'Func_2' with no prototype in function main
Warning dhry_1.c 156: Call to function 'Proc_7' with no prototype in function main
Warning dhry_1.c 161: Call to function 'Proc_8' with no prototype in function main
Warning dhry_1.c 163: Call to function 'Proc_1' with no prototype in function main
Warning dhry_1.c 167: Call to function 'Func_1' with no prototype in function main
Warning dhry_1.c 170: Call to function 'Proc_6' with no prototype in function main
Warning dhry_1.c 171: Call to function 'strcpy' with no prototype in function main
Warning dhry_1.c 172: Conversion may lose significant digits in function main
Warning dhry_1.c 173: Conversion may lose significant digits in function main
Warning dhry_1.c 181: Call to function 'Proc_2' with no prototype in function main
Warning dhry_1.c 266: Call to function 'exit' with no prototype in function main
Warning dhry_1.c 286: Call to function 'Proc_3' with no prototype in function Proc_1
Warning dhry_1.c 295: Call to function 'Proc_6' with no prototype in function Proc_1
Warning dhry_1.c 298: Call to function 'Proc_7' with no prototype in function Proc_1
Warning dhry_1.c 338: Call to function 'Proc_7' with no prototype in function Proc_3
dhry_2.c:
Warning dhry_2.c 44: Call to function 'Func_3' with no prototype in function Proc_6
Warning dhry_2.c 155: Call to function 'Func_1' with no prototype in function Func_2
Warning dhry_2.c 168: Call to function 'strcmp' with no prototype in function Func_2
console.c:
bcppsio.c:
Turbo Assembler  Version 4.0  Copyright (c) 1988, 1993 Borland International

Assembling file:   bcppsio.ASM
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  213k

        tasm /mx /d__s__ /dSTKSIZE=8192 /zd /dPDREMOTE /dFLOAT=2 /e bcpprtl.asm

Turbo Assembler  Version 4.0  Copyright (c) 1988, 1993 Borland International

Assembling file:   bcpprtl.asm
Assembling for the small memory model
Building Paradigm PDREMOTE-compatible library support
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  226k

        tasm /mx /d__s__ /dSTKSIZE=8192 /zd /dPDREMOTE /dFLOAT=2 /e bcppflt.asm

Turbo Assembler  Version 4.0  Copyright (c) 1988, 1993 Borland International

Assembling file:   bcppflt.asm
Assembling for the small memory model
Building Borland C++ emulator floating point support
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  219k

        bcc bcppdmm.c efast.c eleds.c timer.c

Borland C++ 4.5 Copyright (c) 1987, 1994 Borland International
```

```
bcppdmm.c:
Turbo Assembler  Version 4.0  Copyright (c) 1988, 1993 Borland International

Assembling file:   bcppdmm.ASM
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  222k

efast.c:
eleds.c:
timer.c:
Turbo Assembler  Version 4.0  Copyright (c) 1988, 1993 Borland International

Assembling file:   timer.ASM
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  217k

        tlink @MAKE0000.@@@

Turbo Link  Version 7.00 Copyright (c) 1987, 1994 Borland
International
        locate -cdhry21.rt  -DHASFARDATA dhry21

Paradigm LOCATE - Version 5.00a
Copyright (C) 1987-1994 Paradigm Systems.  All rights reserved.

Warning W1032: Segment '_TEXT/CODE' is output to a read/write region
Warning W1032: Segment 'E87_PROG/CODE' is output to a read/write region
Warning W1032: Segment 'EMU_PROG/CODE' is output to a read/write region
Warning W1032: Segment '_INIT_/INITDATA' is output to a read/write region
Warning W1032: Segment '_INITEND_/INITDATA' is output to a read/write region
Warning W1032: Segment '_EXIT_/EXITDATA' is output to a read/write region
Warning W1032: Segment '_EXITEND_/EXITDATA' is output to a read/write region
Warning W1032: Segment '_RD/ROMDATA' is output to a read/write region
Warning W1032: Segment '_DATA/ROMDATA' is output to a read/write region
Warning W1032: Segment '_CVTSEG/ROMDATA' is output to a read/write region
Warning W1032: Segment '_SCNSEG/ROMDATA' is output to a read/write region
Warning W1032: Segment '_ERD/ENDROMDATA' is output to a read/write region
Warning W1032: Segment '_BFD/ROMFARDATA' is output to a read/write region
Warning W1032: Segment '_BRFD/ROMFARDATA' is output to a read/write region
Warning W1032: Segment '_ERFD/ENDROMFARDATA' is output to a read/write region
```

The warnings from Locate indicate that you are using the SRAM area for loading your code. Since it is intended that the program run in SRAM these messages can be ignored.



Running Paradigm's Locate

Operation of this tool is automatic. The supplied make files automatically invoke Locate to change the .EXE file into a located image loadable by the debugger. In the process, you may see that Locate warns you that some or all of your sections are in read/write areas of your target memory. This is completely normal in that your program is being loaded into the SRAM on the SD186EM board.

If Locate warns you that sections or classes overlap, take this seriously! This message is telling you that you haven't allowed enough space between code and data. To correct this problem edit your <your_program_name>.rt file. The entries for class CODE and class DATA need to be changed so that the overlap messages go away. Watch out that you do not push the program to load beyond the SRAMs address range.

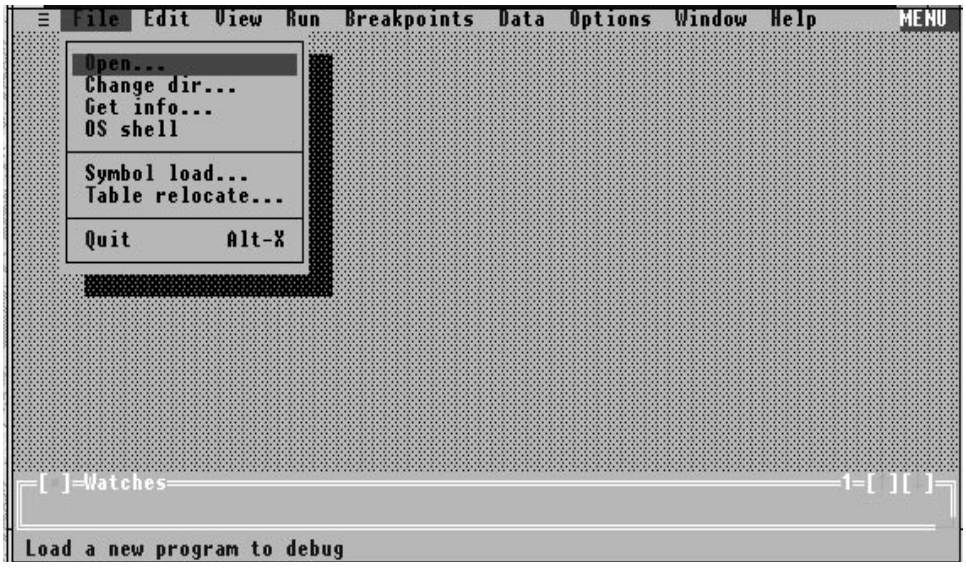
Running Paradigm's Debugger

Invoke the Debugger from the DOS command line. Use the file|open command to request the loading of your program into the SD186EM target.

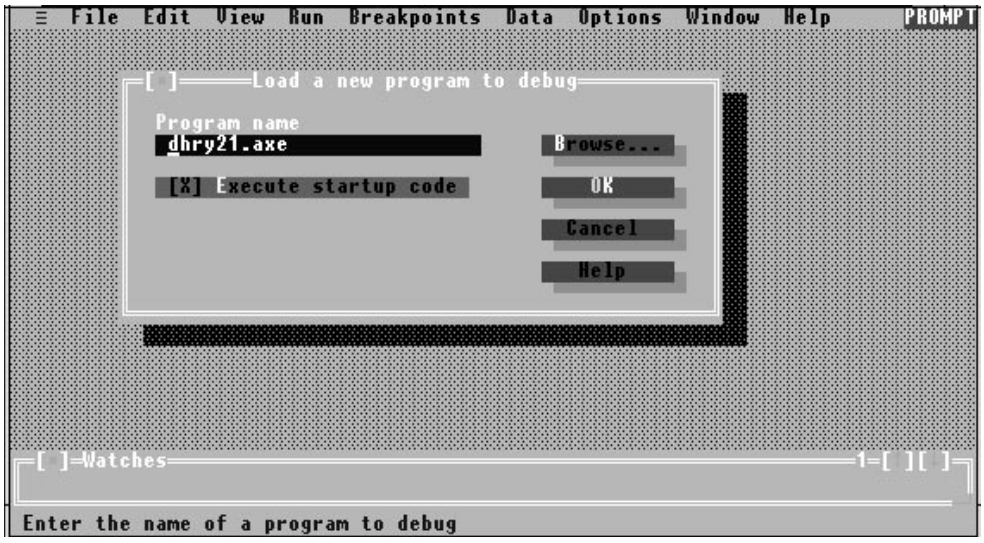
Most programs use printf() and scanf(). To support user input and output you must enable the debuggers '*stdio'* window which allows the target to communicate with you on your monitor and you to send information the target with your PC keyboard.

To operate PDRT186 follow these steps:

At the DOS (or Windows DOS box) command enter the command: "c:\>PDRT186" you should see the following screen after clicking the 'file' menu item.

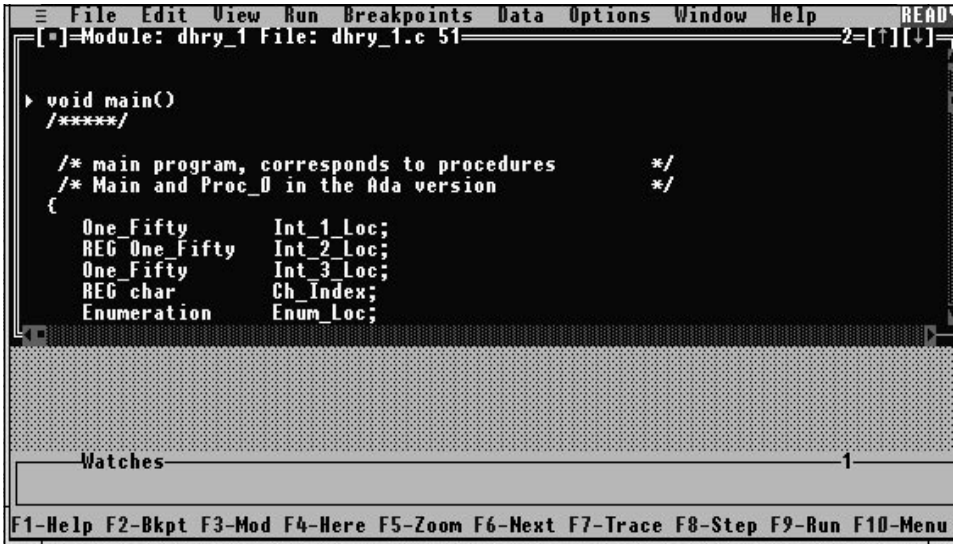


After clicking the '*open'* menu item, you will be asked for the file name.



In the provided input line, enter the name of the program you wish to run. To run the Dhrystone, copy the above information (your current directory must be where the AMD supplied files are found). Then click '*ok'*. Remember the supplied AMD files include a pre-compiled Dhrystone program to test.

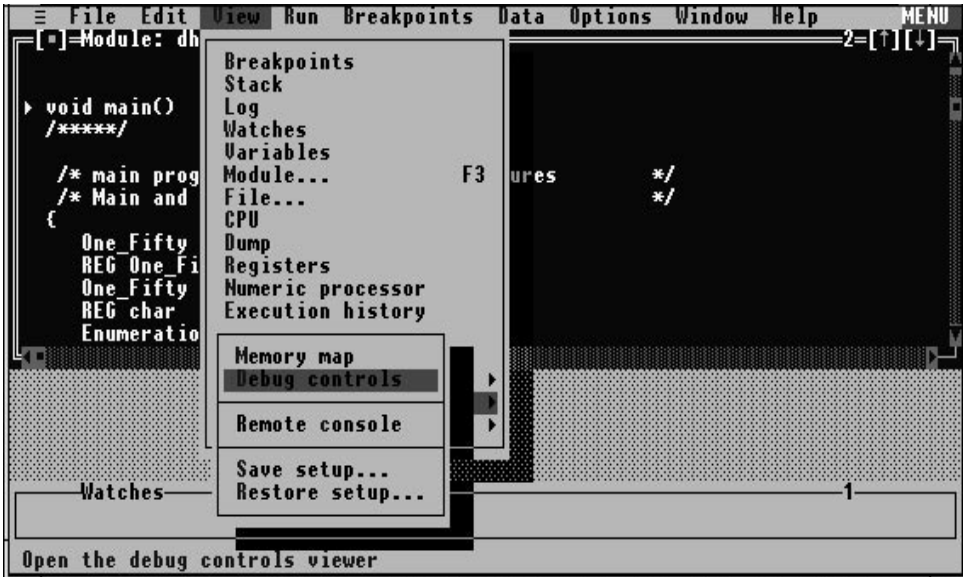
Once the code is loaded into the SD186EM target (Dhrystone shown here) you should see a screen like this showing your source code.



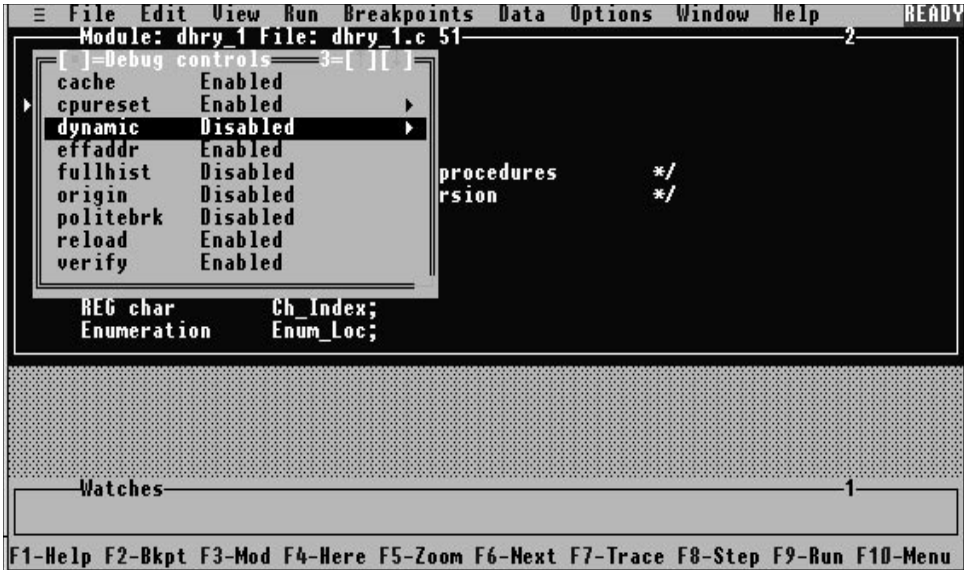
To enable printing to your screen and keyboard communications, select the '*view'* menu and click on '*Paradigm'*.

For programs that need keyboard input do the following. Enabling dynamic mode will slow down execution of the program under test.

Next, click on the '*debug controls'* menu selection.

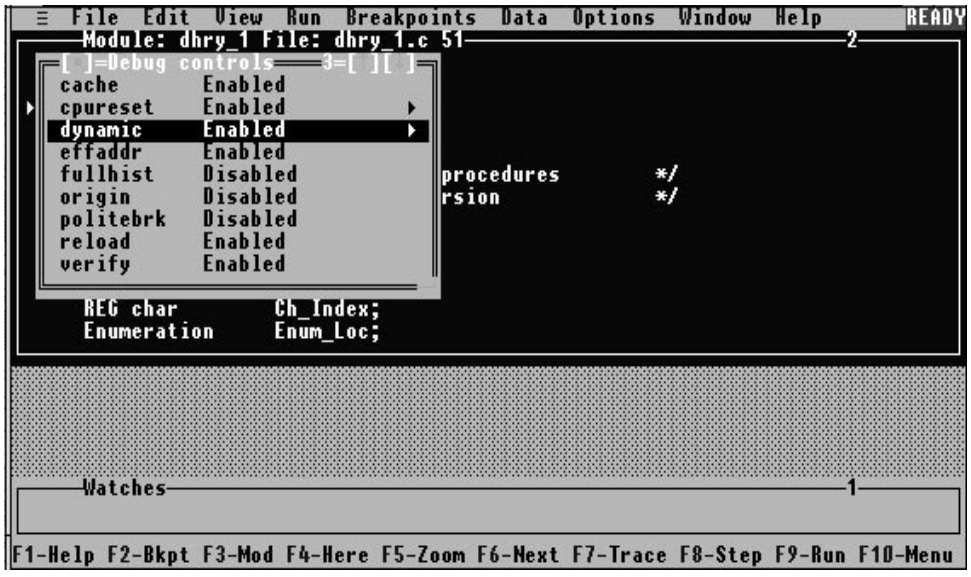


In Paragim's version of Turbo Debugger, they have added a number of features. Of interest is enabling the dynamic option in the menu shown below. As can be seen when powered up the debugger has dynamic mode disabled. Click on '*dynamic'* to make the changes needed to enable communications.



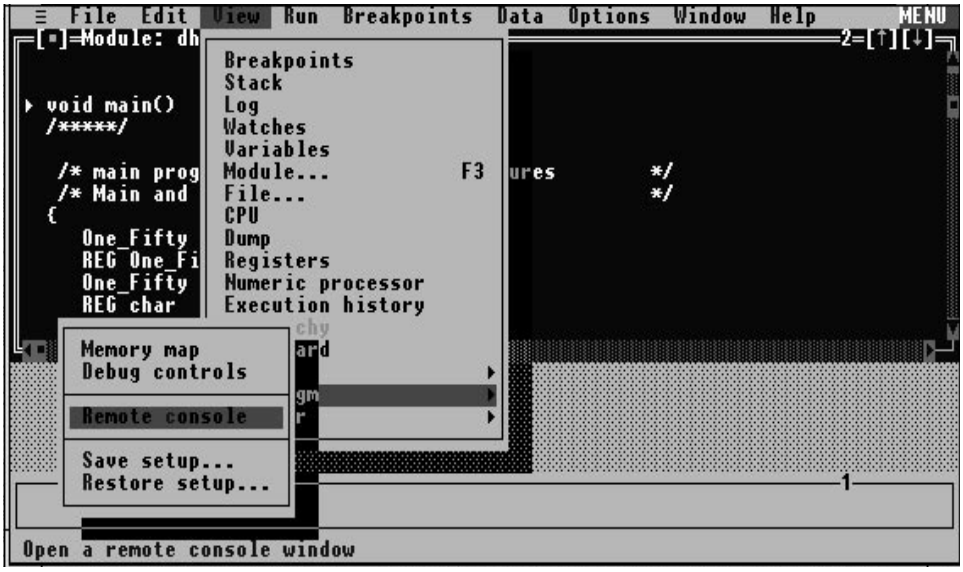
Enabling, the dynamic mode will allow the SD186EM to bi-directionally communicate with the debugger. Make the resulting dialog box, shown below, match yours, then click on *'OK'*.

Once you have entered all the information correctly you should see that dynamic mode is enabled.



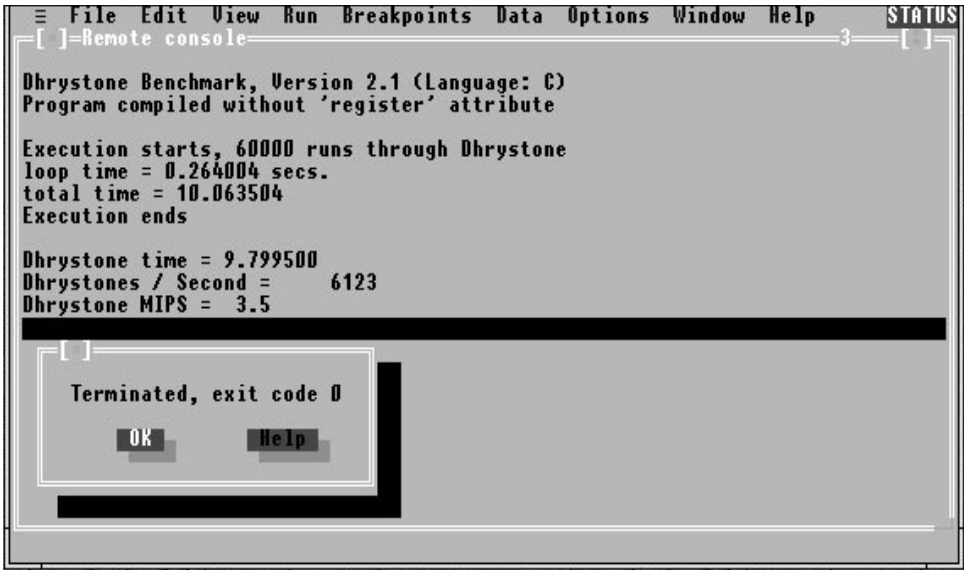
To close the debug control box click on the square in the upper left hand side of the window frame.

You must next open a remote console window. This is the area in which the SD186EM communicates to you and you can send keyboard information to the board. Simply click on the '*remote console'* menu item.



Your screen should now have the remote console window on it.

Click on the arrow pointing upward on the frame of the remote console window. This will make the window enlarge to cover the full screen. Run the benchmark by pressing the 'F9' key on the keyboard. You should see the following results:



After running Dhrystone try editing "paradigm.mkf" changing the '-O2' to '-G'. Also add '-DREG=register to the 'optflags' line. Re-compile the program and run it again. Did it get faster?

Optimization

As you benchmark your programs keep in mind that optimization can greatly improve the performance of the program as well as inflate code size, cause erratic behavior and even cause the program to run slower! Take careful note of the compiler selections you use (-O1, -G, -O2 etc.) as they determine the optimizations the compiler will perform on your code.

Below are some simple explanations of each major optimization.

Be sure you know the effect of each before you attempt to call any timing optimal.

Conventional wisdom says there are three components to generating good code on the 80x86 processors; register allocation, register allocation and register allocation.

Global register allocation

Because memory references are so expensive on these processors, it is extremely important to minimize those references through the intelligent use of registers. Global register allocation both increases the speed and decreases the size of your application. You should always use global register allocation when compiling your application with optimizations on.

Dead-code elimination

Although you may never intentionally write code to do things which are unnecessary, the optimizer may reveal possibilities to eliminate stores into variables that are not needed.

Common subexpression elimination

Common subexpression elimination is the process of finding duplicate expressions within the target scope and storing the calculated value of those expressions once so as to avoid recalculating the expression. Although in theory this optimization could reduce code-size, in practice, it is a speed optimization and will only rarely result in size reductions. You should also use global common subexpression analysis if you like to reuse expressions rather than create explicit stack locations for them.

Loop invariant code motion

Moving invariant code out of loops is a speed optimization. The optimizer uses the information about all the expressions in the function gathered during common subexpression elimination to find expressions whose values do not change inside a loop. To prevent the calculation from being done many times inside the loop, the optimizer moves the code outside the loop so that it is calculated only once. The optimizer then reuses the calculated value inside the loop.

You should use loop invariant code motion whenever you are compiling for speed and you have used global common subexpressions, since moving code out of loops can result in enormous speed gains.

Copy propagation	Propagating copies is primarily speed optimization, but since it never increases the size of your code, it is safe to use if you have enabled **-Og**. Like loop invariant code motion, copy propagation relies on the analysis performed during common subexpression elimination. Copy propagation means that the optimizer remembers the values assigned to expressions and uses those values instead of loading the value of the assigned expressions. Copies of constants, expressions, and variables may be propagated.
Pointer aliasing	Pointer aliasing is not an optimization in itself, but it does affect the way the optimizer performs common subexpression elimination and copy propagation. When pointer aliasing is turned on, it allows the optimizer to maintain copy propagation information across function calls and to maintain common subexpression information across some stores. Otherwise, the optimizer must discard information about copies and subexpressions in these situations.
Induction-variable analysis and strength reduction	Creating induction variables and performing strength reduction are speed optimizations performed on loops. The optimizer uses a mathematical technique called induction to create new variables out of expressions used inside a loop. These variables are called *induction variables.* The optimizer assures that the operations performed on these new variables are computationally less expensive (reduced in strength) than those used by the original variables. Opportunities for these optimizations are common if you use array indexing inside loops, since a multiplication operation is required to calculate the position in the array which is indicated by the index.
Loop compaction	Loop compaction takes advantage of the string move instructions on the 80x86 processors by replacing the code for a loop with such an instruction. Depending on the complexity of the operands, the compacted loop code may also be smaller than the corresponding non-compacted loop. You may wish to experiment with this optimization if you are compiling for size and have loops of this nature.
Structure copy inlining	The most visible optimization performed when compiling for speed as opposed to size is that of inlining structure copies. When you enable **-Ot**, the compiler determines whether it can safely generate code to perform a **rep movsw** instruction instead of calling a helper function to do the copy. For structures and unions of over eight bytes in length, performing this optimization produces faster structure copies than the corresponding helper function call.
Code compaction	The most visible optimization performed when compiling for size is code compaction. In code compaction, the optimizer scans the generated code for duplicate sequences. When such sequences warrant, the optimizer replaces one sequence of code with a jump to the other, thereby eliminating the first piece of code. **switch** statements contain the most opportunities for code compaction.

Redundant load suppression	Load suppression is both a speed and size optimization. When **-Z** is enabled, the optimizer keeps track of the values it loads into registers and suppresses loads of values which it already has in a register.
Intrinsic function inlining	There are times when you would like to use one of the common string or memory functions, such as **strcpy** or **memcmp**, but you do not want to incur the overhead of a function call. By using **-Oi**, the compiler will generate the code for these functions within your function's scope, eliminating the need for a function call. The resulting code will execute faster than a call to the same function, but it will also be larger.
Register parameter passing	The Borland C++ compiler has two calling conventions where parameters are passed in registers instead of on the stack: Object Data and fastcall.

Available Software

File Name Listing

If you have received the diskette containing the file “amdfiles.zip”. This disk is supplied by AMD, you should see the following files after decompressing it:

```
am186em.h          dhry21.loc         dhry_2.c           paradigm.mkf
asmrules.h         dhry21.map         efast.c            support.h
dhry.h             eleds.c            timer.c
dhry21.axe         dhry21.rom         eleds.h            timer.h
dhry21.cfg         dhry21.rt          emon.trm
dhry21.hex         dhry_1.c           makefile
```

In directory “bench”:

```
am186em.h          bench.loc          paradigm.mkf
asmrules.h         bench.map          efast.c
eleds.c            timer.c            support.h
eleds.h            timer.h
bench.cfg          bench.rt           emon.trm
bench.hex          bench.axe          makefile
```

Use and unzip utility to decompress the files. Be sure to use the option to create and decompress sub-directories.

BBS Access

If the disk was not included, or if you would like a newer revision of this material, contact the AMD 29K BBS system by dialing: 1-800-2929-AMD after the automated answering system connects then have your modem signal the number 1 to select the BBS system.

Other Information

Data Sheet

To obtain a data sheet on the Am186EM call, 1-408-749-5703 and ask for PID number 19168.

Other Application Notes

“Building Embedded C-Language Applications for the Am186EM Microcontroller” by Becky Cavanaugh and Melanie Typaldos.

To request this call:

1-800-2929-AMD or your local sales office.

Ordering Board Level Products

You may order an SD186EM development board through your local AMD office. If you need the local number call: 1-408-732-2400.

Books on the Subject

“Mastering Turbo Debugger” by Tom Swan ISBN # 0-672-48454-4

Hayen Books

Appendix A

Understanding the Dhrystone Benchmark

The included text was taken from the source code of the Dhrystone benchmark. Of interest is the information about the testing done on the benchmark. This data helps the programmer determine the amount of similarity between the benchmark and the actual product code. Programs with a large degree of similar operations and memory utilization should see matching performance gains.

```
/*****************************************************************
 *
 *                    "DHRYSTONE" Benchmark Program
 *                    -----------------------------
 *
 *  Version:    C
 *
 *  Author:     Reinhold P. Weicker
 *                      Siemens AG, E STE 35
 *                      Postfach 3240
 *                      8520 Erlangen
 *                      Germany (West)
 *                              Phone:  [xxx-49]-9131-7-20330
 *                                      (8-17 Central European Time)
 *                              Usenet: ..!mcvax!unido!estevax!weicker
 *
Original Version (in Ada) published in   "Communications of the ACM"
vol. 27., no. 10 (Oct. 1984), pp. 1013 - 1030, together with the
statistics on which the distribution of statements etc. is based.  In
this C version, the following C library functions are used:
 - strcpy, strcmp (inside the measurement loop)
 - printf, scanf (outside the measurement loop)
In addition, UNIX system calls "times ()" or "time ()" are used for
execution time measurement. For measurements on other systems, these
calls have to be changed.
Collection of Results:
Reinhold Weicker (address see above) and Rick Richardson
PC Research. Inc.
94 Apple Orchard Drive
Tinton Falls, NJ 07724
Phone:  (201) 834-1378 (9-17 EST)
Usenet: ...!seismo!uunet!pcrat!rick

Please send results to Reinhold Weicker and/or Rick Richardson.
- Complete information should be given on hardware and software used.
- Hardware information includes: Machine type, CPU, type and size of
caches; for microprocessors: clock frequency, memory speed (number of
wait states).
- Software information includes: Compiler (and runtime library)
manufacturer and version, compilation switches, OS version.
- The Operating System version may give an indication about the
compiler; Dhrystone itself performs no OS calls in the measurement loop.

  History:    This version has been made for two reasons:

              1) There is an obvious need for a common C version of
              Dhrystone, since C is at present the most popular system
              programming language for the class of processors
```

(microcomputers, minicomputers) where Dhrystone is used most.
There should be, as far as possible, only one C version of Dhrystone such that results can be compared without restrictions. In the past, the C versions distributed by Rick Richardson (Version 1.1) and by Reinhold Weicker had small (though not significant) differences.

2) As far as it is possible without changes to the Dhrystone statistics, optimizing compilers should be revented from removing significant statements.

This C version has been developed in cooperation with Rick Richardson (Tinton Falls, NJ), it incorporates many ideas from the "Version 1.1" distributed previously by him over the UNIX network Usenet.
I also thank Chaim Benedelac (National Semiconductor), David Ditzel (SUN), Earl Killian and John Mashey (MIPS), Alan Smith and Rafael Saavedra-Barrera (UC at Berkeley) for their help with comments on earlier versions of the benchmark.

Changes: In the initialization part, this version follows mostly Rick Richardson's version distributed via Usenet, not the version distributed earlier via floppy disk by Reinhold Weicker.
As a concession to older compilers, names have been made unique within the first 8 characters.
Inside the measurement loop, this version follows the version previously distributed by Reinhold Weicker.

At several places in the benchmark, code has been added, but within the measurement loop only in branches that are not executed. The intention is that optimizing compilers should be prevented from moving code out of the easurement loop, or from removing code altogether. Since the statements that are executed within the measurement loop have NOT been changed, the numbers defining the "Dhrystone distribution"(distribution of statements, operand types and locality) still hold. Except for sophisticated optimizing compilers, execution times for this version should be the same as for previous versions.

Since it has proven difficult to subtract the time for the measurement loop overhead in a correct way, the loop check has been made a part of the benchmark. This does have an impact - though a very minor one - on the distribution statistics which have been updated for this version.

All changes within the measurement loop are described and discussed in the companion paper "Rationale for Dhrystone version 2".

Because of the self-imposed limitation that the order and distribution of the executed statements should not be changed, there are still cases where optimizing compilers may not generate code for some statements. To a certain degree, this is unavoidable for small synthetic benchmarks.
Users of the benchmark are advised to check code listings whether code is generated for all statements of Dhrystone.

```
*************************************************************************

 Defines:     The following "Defines" are possible:
              -DREG=register          (default: Not defined)
              As an approximation to what an average C programmer
              might do, the "register" storage class is applied
              (if enabled by -DREG=register)
              - for local variables, if they are used (dynamically)
              five or more times
              - for parameters if they are used (dynamically) six or
              more times
              Note that an optimal "register" strategy is compiler-
              dependent, and that "register" declarations do not
              necessarily lead to faster execution
              -DNOSTRUCTASSIGN        (default: Not defined)
              Define if the C compiler does not support assignment of
              structures.
              -DNOENUMS               (default: Not defined)
              Define if the C compiler does not support enumeration
              types.
              -DTIMES                 (default)
              -DTIME
              The "times" function of UNIX (returning process times)
                    or the "time" function (returning wallclock time)
                      is used for measurement.
                      For single user machines, "time ()" is adequate.
              For multi-user machines where you cannot get single-user
                      access, use the "times ()" function. If you have
                      neither, use a stopwatch in the dead of night.
                      "printf"s are provided marking the points "Start
                  Timer" and "Stop Timer". DO NOT use the UNIX
                  "time(1)" command, as this will measure the
                  total time to
                    run this program, which will (erroneously) include
                      the time to allocate storage (malloc) and to
                        perform the initialization.
              -DHZ=nnn                (default: 60)
                      The function "times" returns process times in
                      1/HZ seconds, with HZ = 60 for most systems.
                      CHECK YOUR SYSTEM DESCRIPTION BEFORE YOU JUST
                        APPLY THE DEFAULT VALUE.

*************************************************************************

  Compilation model and measurement (IMPORTANT):

  This C version of Dhrystone consists of three files:
  - dhry_global.h (this file, containing global definitions and comments)
  - dhry_pack_1.c
  - dhry_pack_2.c

  The following "ground rules" apply for measurements:
  - Separate compilation
  - No procedure merging
  - Otherwise,compiler optimizations are allowed but should be indicated
  - Default results are those without register declarations
See the companion paper "Rationale for Dhrystone Version 2" for a more
  detailed discussion of these ground rules.

For 16-Bit processors (e.g. 80186, 80286), times for all compilation
models ("small", "medium", "large" etc.) should be given if possible,
together with a definition of these models for the compiler system used.
```

```
************************************************************************

  Dhrystone (C version) statistics:

 The following program contains statements of a high level programming
  language (here: C) in a distribution considered representative:

    assignments                  52 (51.0 %)
    control statements           33 (32.4 %)
    procedure, function calls    17 (16.7 %)

103 statements are dynamically executed. The program is balanced with
  respect to the three aspects:

    - statement type
    - operand type
    - operand locality
         operand global, local, parameter, or constant.

The combination of these three aspects is balanced only approximately.

  1. Statement Type:
  -----------------              number   Total

     V1 = V2                      9
       (incl. V1 = F(..)
     V = Constant                12
     Assignment,                  7
      with array element
     Assignment,                  6
       with record component
                                 --
                                 34       34

     X = Y +|-|"&&"|"|" Z         5
     X = Y +|-|"==" Constant      6
     X = X +|- 1                  3
     X = Y *|/ Z                  2
     X = Expression,              1
           two operators
     X = Expression,              1
           three operators
                                 --
                                 18       18

     if ....                     14
       with "else"                7
       without "else"             7
           executed               3
           not executed           4
     for ...                      7  |  counted every time
     while ...                    4  |  the loop condition
     do ... while                 1  |  is evaluated
     switch ...                   1
     break                        1
     declaration with             1
       initialization
                                 --
                                 34       34

     P (...)  procedure call     11
       user procedure            10
       library procedure          1
     X = F (...)
```

function call	6	
user function	5	
library function	1	
	--	
	17	17

		103

The average number of parameters in procedure or function calls is 1.82 (not counting the function values as implicit parameters).

2. Operators

	number	approximate percentage
Arithmetic	32	50.8
+	21	33.3
-	7	11.1
*	3	4.8
/ (int div)	1	1.6
Comparison	27	42.8
==	9	14.3
/=	4	6.3
>	1	1.6
<	3	4.8
>=	1	1.6
<=	9	14.3
Logic	4	6.3
&& (AND-THEN)	1	1.6
| (OR)	1	1.6
! (NOT)	2	3.2
	--	-----
	63	100.1

3. Operand Type (counted once per operand reference):

	number	approximate percentage
Integer	175	72.3 %
Character	45	18.6 %
Pointer	12	5.0 %
String30	6	2.5 %
Array	2	0.8 %
Record	2	0.8 %
	---	-------
	242	100.0 %

When there is an access path leading to the final operand (e.g. a record component), only the final data type on the access path is counted.

4. Operand Locality:

	number	approximate

		percentage
local variable	114	47.1 %
global variable	22	9.1 %
parameter	45	18.6 %
value	23	9.5 %
reference	22	9.1 %
function result	6	2.5 %
constant	55	22.7 %
	---	-------
	242	100.0 %

The program does not compute anything meaningful, but it is syntacticallyand semantically correct. All variables have a value assigned to them before they are used as a source operand.

There has been no explicit effort to account for the effects of a cache, or to balance the use of long or short displacements for code or data.

Appendix B

Summary of Processors

Below is a summary of computer systems and their processors running both Dhrystone 1.1 and Dhrystone 2.1.

The Header test defines the data and explains how an updated version of this report can be obtained. Dhrystone 2.1 and 1.1 MIPS Results (Language: C). The Dhrystone programs are by Reinhold Weicker. His email address is: weicker.muc@sni.de (in general), or weicker.muc@sni-usa.com (from North America).

Dhrystone is a short synthetic benchmark program intended to be representative for system (integer) programming. Based on published statistics on use of programming language features: see original publication in CACM 27,10 (Oct 1984). Orginally published in ADA, now mostly used in C. Version 2 (in C) published in SIGPLAN Notices 23,8 (Aug 1988), together with measurement rules. Version 1 is no longer recommended since state-of-the-art compilers can eliminate too much 'dead code' from the benchmark (However, quoted MIPS numbers are often based on version 1). Problems: Due to its small size (100 HLL statements, 1-1.5 KB code), the memory system outside the cache is not tested; compilers can too easily optimize for Dhrystone; string operations are somewhat over-represented.

Recommendation: Use it for controlled experiments only; don't blindly trust single Dhrystone MIPS numbers quoted somewhere (don't do this for any benchmark). The Dhrystone C Programs (dhry.shar), and latest table of results (dhry.tbl) are available via anonymous ftp from 'ftp.nosc.mil' in directory 'pub/aburto'. The ftp.nosc.mil IP address is: 128.49.192.51 .

Please send new results (systems, machines, compilers, compiler options) to 'aburto@marlin.nosc.mil'. I will keep results updated, post to 'comp.benchmarks' periodically, and send results to Reinhold Weicker.

Please read the 'dhry.doc' file first for information regarding running the programs and submitting results. If you need the individual files please send email to aburto@marlin.nosc.mil . These Dhrystone source and results are also available via ftp from netlib@ornl.gov .
The VAX 11/780 MIPS reference is 1757 for both Dhrystone V1.1 and V2.1. I used the same VAX 11/780 MIPS reference (1757) for both V1.1 and V2.1 since the results for unoptimized code in general were very similar, if not identical, in this case. Reinhold Weicker recommended that both V1.1 and V2.1 MIPS results be collected in this table as this would give a reference on how much more V1.1 can be optimized relative to V2.1. When quoting Dhrystone VAX MIPS ratings it is preferrable to use the V2.1 numbers. Reinhold Weicker has asked me to add the following statement: "Dhrystone author says: Relying on MIPS V1.1 numbers can be hazardous to your professional health" ... Some

compiler results which appear 'too high' compared to other compiler results on the same machine are indicated by an '*' --- just my way of keeping track of exceptional results from certain compilers.

Results as of 05 Nov 1994:

###	System	OS/Compiler	CPU	CPU (MHz)	MIPS V1.1	MIPS V2.1	REF
001	DEC 10000/610 AXP	OpenVMS V1.0	DEC 21064	200.0	194.9	214.8	6
002	DEC 7000/600 AXP	OSF/1 V1.3a	DEC 21064	200.0	195.5	203.3	33
003	DEC 7000/610 AXP	OpenVMS V1.0	DEC 21064	182.0	177.3	195.6	6
004	DEC 3000/800 AXP	OSF/1 V1.3a	DEC 21064	200.0	192.9	189.7	33
005	DEC 4000/710 AXP	OSF/1 V1.3a	DEC 21064	190.0	188.7	189.7	33
006	DEC 4000/610 AXP	OpenVMS V1.0	DEC 21064	160.0	159.0	173.0	6
007	DEC 3000/600 AXP	OSF/1 V1.3a	DEC 21064	175.0	168.5	167.4	33
008	DEC 3000/500 AXP	OpenVMS V1.0	DEC 21064	150.0	146.7	160.1	6
009	HP 9000/735	HP-UX 9.01	PA-RISC7100	99.0	------	146.5	52
010	HP 9000/735	HP-UX 9.01	PA-RISC7100	99.0	------	143.0	46
011	DEC 3000/400 AXP	OpenVMS V1.0	DEC 21064	133.0	129.9	142.1	6
012	DEC 3000/500 AXP	OSF/1 T1.3-3	DEC 21064	150.0	211.0	137.1*	25
013	DEC 3000/500 AXP	OSF/1 T1.3-3	DEC 21064	150.0	157.1	133.7	25
014	DEC 2000/300 AXP	OSF/1 V1.3a	DEC 21064	150.0	140.9	129.4	33
015	DEC 4000/710	OSF	DEC 21064	190.0	186.	126.	51
016	Gateway P5-90	DOS 6.2	Pentium	90.0	172.5	124.4	53
017	IBM RS/6000 590	AIX 3.2.5	Power2	-----	134.2	124.3	47
018	HP 9000/735	HP-UX 9.01	PA-RISC7100	99.0	------	116.6	32
019	PowerMac 8100/80	System 7.1.2	PowerPC 601	80.0	157.7	113.3	56
020	HP 9000/735	HP-UX 9.0	PA-RISC7100	99.0	122.7	110.5	5
021	HP 9000/735	HP-UX 9.01	PA-RISC7100	99.0	122.7	110.1	8
022	SGI Indigo2 Extreme	Irix 4.0.5H	R4000	100.0	------	108.9	60

023	HP 9000/735	HP-UX 9.01	PA-RISC7100	99.0	118.1	106.6	8
024	Intel Pentium/66	UNIX SVR 4.2	Pentium	66.0	111.6	101.2	30
025	HP 9000/755	HP-UX 9.01	PA-RISC7100	99.0	112.0	99.2	55
026	HP 9000/735	HP-UX 9.01	PA-RISC7100	99.0	106.2	97.1	8
027	SGI Indigo2 Extreme	Irix 4.0.5H	R4000	100.0	------	95.3	60
028	HP 9000/755	HP-UX 9.01	PA-RISC7100	99.0	108.4	95.2	55
029	HP 9000/735	HP-UX 9.0	PA-RISC7100	99.0	104.2	95.0	8
030	HP 9000/750	HP-UX 9.05	PA7000	66.0	110.7	94.2	61
031	PowerMac 7100/66	System 7.1.2	PowerPC 601	66.0	129.6	93.1	56
032	ALR Pentium/60	MS DOS 5.0	Pentium	60.0	112.9	91.4	29
033	Intel Pentium/60	UNIX SVR 4.2	Pentium	60.0	100.0	91.4	30
034	HP 9000/712	HP-UX 9.05	PA7100LC	60.0	118.6	88.7	61
035	IBM RS/6000 250	AIX 3.2.5	PowerPC 601	66.0	96.1	83.6	54
036	IBM RS/6000 250	AIX 3.2.5	PowerPC 601	66.0	96.1	82.8	54
037	Sun SPARCserver 20/612	Solaris 2.3	SuperSPARC	60.0	93.6	82.2	58
038	IBM RS/6000 Model 340	AIX 3.2	Power Risc	33.0	------	76.3	60
039	Gateway Pentium P5-90	LINUX 1.1.35	Pentium	90.0	80.6	75.9	59
040	DATEL Pentium P5-90	MS DOS 6.22	Pentium	90.0	73.0	70.0	62
041	ZEOS Pentium P5-90	MS DOS 6.22	Pentium	90.0	72.5	70.0	62
042	HP 9000/750	HP-UX 9.0	PA-RISC	66.0	76.1	69.6	5
043	SGI Indigo Elan	Irix 4.0.5H	R4000	100.0	78.8	69.2	43
044	SGI Crimson Elan	Irix 4.0.5	R4000	50.0	77.5	69.2	43
045	HP 9000/730	HP-UX 9.0	PA-RISC	66.0	76.1	69.2	5
046	SPARCcenter 1000	Solaris 2.3	SuperSPARC	50.0	77.8	68.5	50
047	SGI Indigo2 Extreme	Irix 4.0.5H	R4000	100.0	76.8	67.4	43
048	SPARCstation 10/51	SunOS 4.1.3	SuperSPARC	50.0	74.9	67.0	49
049	SGI Indy PC	Irix 5.1.1	R4000	100.0	65.1	67.0	43
050	HP 9000/730	HP-UX 8.07	PA-RISC	66.0	------	65.2	3
051	IBM RS/6000 Model 550	AIX 3.2.2	Power RISC	42.7	80.8	62.9	2
052	SPARCstation 10/41	SunOS 4.1.3	SuperSPARC	40.0	63.7	58.3	26

053	HP 9000/715	HP-UX 9.0	PA-RISC7100	50.0	62.3	55.2	5
054	SPARCstation Voyager	Solaris 2.3	uSPARC II	60.0	61.5	54.8	50
055	IBM RS/6000 Model 550	AIX 3.2.2	Power RISC	42.7	61.7	54.7	2
056	HP 9000/735	HP-UX 9.01	PA-RISC7100	99.0	53.3	53.4	8
057	SPARCstation 10/41	Solaris 2.3	SuperSPARC	40.0	59.2	52.7	49
058	SPARCstation 10/30	SunOS 4.1.3	SuperSPARC	36.0	58.0	52.7	26
059	HP 9000/720	HP-UX 9.0	PA-RISC	50.0	57.4	52.7	5
060	SPARCstation 10/51	SunOS 4.1.3	SuperSPARC	50.0	55.2	52.6	49
061	HP 9000/710	HP-UX 9.0	PA-RISC	50.0	57.4	52.5	5
062	Intel 486DX2/66	UNIX SVR 4.2	80496DX2	66.0	54.0	51.6*	30
063	SPARCstation 2 (80)	SunOS 4.1.2	Weitek	80.0	50.1	51.3	41
064	SPARCstation 10/41	SunOS 4.1.3	SuperSPARC	40.0	64.7	51.2	26
065	HP 9000/715t	HP-UX 9.05	PA7100	33.0	60.3	50.5	61
066	HP 9000/705	HP-UX 9.05	PA7000	35.0	58.3	49.5	61
067	ALR Evolution V/1	OS/2 2.1	Pentium	60.0	52.3	49.0	38
068	HP 9000/715	HP-UX 9.0	PA-RISC7100	50.0	------	47.9	7
069	SPARCstation 10/30	SunOS 4.1.3	SuperSPARC	36.0	58.7	46.5	26
070	SPARCstation 10/20	SunOS 4.1.2	SuperSPARC	33.	37.5	43.5	13
071	SPARCstation 10/30	SunOS 4.1.3	SuperSPARC	36.0	56.2	42.8	17
072	IBM RS/6000 53H	AIX 3.2.5	RISC	33.	44.9	42.7	54
073	SPARCstation 10/20	SunOS 4.1.2	SuperSPARC	33.	43.2	42.5	13
074	SPARCstation 10/30	SunOS 4.1.3	SuperSPARC	36.0	54.6	40.9	17
075	SPARCstation 10/30	SunOS 4.1.3	SuperSPARC	36.0	43.4	37.9	17
076	SPARCstation LX	Solaris 2.2	SPARC	0.0	45.7	37.7	26
077	ISA/VLB AT Clone	MS DOS 6.0	80486DX2	66.7	41.1	37.7	34
078	SPARCstation 10/30	SunOS 4.1.3	SuperSPARC	36.0	39.3	37.4	21
079	HP 9000/705	HP-UX 9.0	PA-RISC7100	35.0	40.0	36.7	5
080	HP 9000/715	HP-UX 9.0	PA-RISC7100	33.0	41.4	36.3	5
081	486DX2/72	LINUX0.99.15	80486DX2	72.0	38.3	35.4	48
082	SPARCstation LX	Solaris 2.2	SPARC	0.0	44.5	34.2	26

083	Intel 486DX/33	----------	80486DX	33.3	35.7	33.5*	29
084	IBM RS/6000 Model 530	AIX 3.1.5	Power RISC	25.0	37.3	32.3	10
085	IBM RS/6000 Model 530	----------	Power RISC	25.0	37.1	32.2	5
086	Cornell EISA/ISA/VLB	LINUX 1.1.23	80486DX2	66.7	35.3	32.2	57
087	LIJN 486DX2/66	LINUX0.99pl6	80486DX2	66.7	------	31.3	22
088	Gateway 486DX2/66	LINUX 0.99	80486DX2	66.7	------	30.9	3
089	Intel 486DX2/66	OS/2 2.1	80486DX2	66.7	32.9	30.2	35
090	DECstation 5000/133	Ultrix 4.3	R3000	33.	31.6	29.0	12
091	IBM RS/6000 Model 550	AIX 3.2.2	Power RISC	42.7	28.5	28.8	2
092	SPARCstation 2 (40)	SunOS 4.1.3	SPARC	40.0	32.1	28.7	42
093	AT Clone	MS DOS 5.0	80486DX	33.3	33.6	28.3*	15
094	----------	4.3 BSD	R3000	33.	------	28.3	37
095	SPARCstation 2 (50)	SunOS 4.1.3	SPARC	50.0	38.5	27.5	42
096	SGI Indigo XZ	Irix 4.0.5H	R3000	33.	31.0	27.3	43
097	SPARCstation 2 (40)	SunOS 4.1.3	SPARC	40.0	31.0	27.3	26
098	SPARCstation 2 (40)	SunOS 4.1.3	SPARC	40.0	30.1	27.2	26
099	DECstation 5000/133	Ultrix 4.3	R3000	33.	29.3	26.9	12
100	DECstation 5000/133	Ultrix 4.3	R3000	33.	30.0	26.6	12
101	DECstation 5000/133	Ultrix 4.3	R3000	33.	29.5	26.6	12
102	Intel 486DX/33	UNIX SVR 4.2	80486DX	33.3	27.6	26.3*	30
103	IBM RS/6000 Model 320	----------	Power RISC	20.0	29.5	25.8	5
104	Sun SPARCserver 690MP	SunOS 4.1.2	SPARC	40.0	27.9	25.2	2
105	AMI 486DX2/66, EISA	MS DOS 5.0	80486DX2	66.7	28.0	25.2	9
106	AMI 486DX2/66, EISA	MS DOS 5.0	80486DX2	66.7	27.5	25.2	9
107	AMI 486DX2/66, EISA	MS DOS 5.0	80486DX2	66.7	28.0	25.1	9
108	SPARCstation 670	SunOS 4.1.3	SPARC	40.0	27.8	25.1	45
109	AT Clone	MS DOS 5.0	80486DX	33.3	28.3	25.0*	15
110	AMI 486DX2/66, EISA	MS DOS 5.0	80486DX2	66.7	28.0	25.0	9
111	PT-SYS5K	Sun4.1.3B-T1	LSI SPARC	40.0	28.9	24.9	16
112	Interpro 2430	Sys V R3.1	----------	0.0	------	24.8	36

113	Insight 486DX/50, ISA	IBM OS/2 2.1	80486DX	50.0	25.7	24.0	28
114	DECstation 5000/133	Ultrix 4.3	R3000	33.	26.4	23.8	12
115	Insight 486DX/50, ISA	IBM OS/2 2.0	80486DX	50.0	25.3	23.7	18
116	SPARCstation 2 (40)	SunOS 4.1.2	SPARC	40.0	26.3	23.6	2
117	Self-Assembled PC	386BSD 0.1	80486DX	50.0	25.4	23.4	24
118	Hitachi EX/60	OSF/1 V1.0.0	Hitachi 370	0.0	23.1	22.9	39
119	MIPS 3230	----------	R3000	25.0	24.5	22.2	5
120	DECstation 5000/200	----------	R3000	25.0	24.2	22.1	5
121	Amiga 2000/FusionForty	AmigaDOS 2.1	68040	28.0	23.3	21.6	23
122	Sun SPARCserver 690MP	SunOS 4.1.2	SPARC	40.0	23.7	21.3	2
123	SPARCstation 2 (40)	SunOS 4.1.3	SPARC	40.0	19.8	21.3	42
124	Sun SPARCserver 690MP	SunOS 4.1.2	SPARC	40.0	21.1	20.9	2
125	SPARCstation 2 (40)	----------	SPARC	40.0	28.5	20.3	5
126	SPARCstation 670	SunOS 4.1.3	SPARC	40.0	21.7	20.1	45
127	Amiga 2000/FusionForty	AmigaOS 2.1	68040	33.3	23.3	19.5	20
128	NeXTstation Turbo	NeXTSTEP 3.0	68040	33.3	20.0	19.5	11
129	80486DX/33	SCO UNIX 3.2	80486DX	33.3	------	19.3	60
130	Interpro 2430	Sys V R3.1	----------	0.0	------	19.2	36
131	DECstation 5000/133	Ultrix 4.3	R3000	33.	19.5	19.0	12
132	NeXTstation/33 Turbo	NeXTSTEP 3.0	68040	33.3	20.0	18.9	11
133	Amiga 4000/25	AmigaOS	68040	25.0	22.2	18.7	4
134	CLUB Falcon 433, ISA	MS DOS 5.0	80486DX	33.3	20.4	18.6	9
135	CLUB Falcon 433, ISA	MS DOS 5.0	80486DX	33.3	20.3	18.6	9
136	LAN Technologies	ESIX 4.0.3A	80486DX	33.3	20.2	18.6	14
137	CLUB Falcon 433, ISA	MS DOS 5.0	80486DX	33.3	20.2	18.6	9
138	LAN Technologies	ESIX 4.0.3A	80486DX	33.3	20.4	18.1	14
139	LAN Technologies	ESIX 4.0.3A	80486DX	33.3	20.1	18.1	14
140	Vega 486DX/33	MS DOS 5.0	80486DX	33.3	19.8	18.1	2
141	Vega 486DX/33	MS DOS 5.0	80486DX	33.3	19.7	18.1	2
142	LAN Technologies	ESIX 4.0.3A	80486DX	33.3	20.1	18.0	14

143	Vega 486DX/33	MS DOS 5.0	80486DX	33.3	19.4	17.9	45
144	LAN Technologies	ESIX 4.0.3A	80486DX	33.3	20.3	17.8	14
145	Vega 486DX/33	MS DOS 5.0	80486DX	33.3	19.6	17.6	45
146	OPUS 5120, SPARC 1+	SunOS 4.1.2	SPARC 7	25.0	19.3	17.4	40
147	AMI 486DX2/66, EISA	MS DOS 5.0	80486DX2	66.7	20.6	17.3	9
148	AMI 486DX2/66, EISA	MS DOS 5.0	80486DX2	66.7	18.8	16.9	9
149	AMI 486DX/33, ISA	LINUX0.99p17	80486DX	33.3	18.4	16.9	19
150	Sun SPARCstation 1+	SunOS 4.1.3	SPARC	25.0	18.6	16.8	31
151	Vega 486DX/33	MS DOS 5.0	80486DX	33.3	18.2	16.8	2
152	Vega 486DX/33	MS DOS 5.0	80486DX	33.3	18.4	16.7	2
153	Vega 486DX/33	MS DOS 5.0	80486DX	33.3	17.9	16.3	2
154	Vega 486DX/33	MS DOS 5.0	80486DX	33.3	17.9	16.3	2
155	Vega 486DX/33	MS DOS 5.0	80486DX	33.3	17.8	16.3	2
156	SGI Personal Iris 25G	Irix 4.0.5H	R3000	20.0	17.4	15.6	43
157	OPUS 5120, SPARC 1+	SunOS 4.1.2	SPARC 7	25.0	16.0	14.3	40
158	OPUS 5120, SPARC 1+	SunOS 4.1.2	SPARC 7	25.0	15.9	14.2	40
159	OPUS 5120, SPARC 1+	SunOS 4.1.2	SPARC 7	25.0	15.8	14.2	40
160	LAN Technologies	ESIX 4.0.3A	80486DX	33.3	14.0	13.8	14
161	LAN Technologies	ESIX 4.0.3A	80486DX	33.3	13.9	13.8	14
162	DECstation 3100	----------	R3000	16.7	14.9	13.4	5
163	Sun SPARCstation 1+	SunOS 4.1.2	SPARC	25.0	14.7	13.2	2
164	Amdahl 580	UTS 5.0 R1.2	----------	-----	13.1	-----	0
165	Sun IPC	----------	SPARC	25.0	17.6	13.0	5
166	CLUB Falcon 433, ISA	MS DOS 5.0	80486DX	33.3	13.5	12.2	9
167	Sun SPARCstation 1+	SunOS 4.1.2	SPARC	25.0	11.8	12.1	2
168	CLUB Falcon 433, ISA	MS DOS 5.0	80486DX	33.3	12.2	11.9	9
169	Vega 486DX/33	MS DOS 5.0	80486DX	33.3	12.6	11.8	2
170	Vega 486DX/33	MS DOS 5.0	80486DX	33.3	12.1	11.7	2
171	80386	SCO UNIX 3.2	80386	40.0	------	11.5	60
172	Vega 486DX/33	MS DOS 5.0	80486DX	33.3	12.3	11.3	2

173	Vega 486DX/33	MS DOS 5.0	80486DX	33.3	11.6	11.2	2
174	Sun 4/280	SunOS 4.1.2	MB86900	16.7	12.5	10.5	17
175	Sun SPARCserver 690MP	SunOS 4.1.2	SPARC	40.0	13.1	10.4	2
176	SPARCstation 2 (40)	SunOS 4.1.2	SPARC	40.0	12.6	9.8	2
177	VAX 8650	4.3 BSD	----------	18.0	6.3	6.2	2
178	Sun SPARCstation 1+	SunOS 4.1.2	SPARC	25.0	7.0	5.9	2
179	VAX 8650	4.3 BSD	----------	18.0	5.4	5.4	2
180	VAX 8650	4.3 BSD	----------	18.0	5.0	5.0	2
181	Sony NWS-1750	----------	68030	25.0	4.8	4.6	27
182	Amiga 2000/G-Force	NetBSD 0.9a	68030	25.0	5.0	4.4	44
183	VAX 8600	VMS	----------	-----	4.1	-----	0
184	Amiga 2000/CSA-MMR	AmigaOS 1.3	68030	33.3	3.4	3.6	2
185	Amiga 2000/CSA-MMR	AmigaOS 1.3	68030	33.3	3.3	3.4	2
186	Amiga 2000/CSA-MMR	AmigaOS 1.3	68030	33.3	3.0	3.1	2
187	MASSCOMP 5600/5700	RTU V3.0	68020	16.7	2.7	-----	0
188	Gould PN9080	UTX-32 1.1c	----------	-----	2.6	-----	0
189	Atari Mega ST 4	MiNT/TOS1.04	68030	25.0	2.2	2.6	40
190	Atari Mega ST 4	MiNT/TOS1.04	68030	25.0	2.2	2.6	40
191	Sun 3/160C	Sun 3.0	68020	16.7	2.4	-----	0
192	MASSCOMP 5400	RTU V3.0	68020	16.7	2.3	-----	0
193	Sun 3/180	Sun 4.2	68020	16.7	2.2	-----	0
194	Sun 3/75	Sun 4.2 V3	68020	16.7	2.0	-----	0
195	Pyramid 90x	OSx 2.5	----------	-----	1.8	-----	0
196	Pyramid 90x	OSx 2.3	----------	-----	1.3	-----	0
197	VAX 11/785	VMS	----------	-----	1.2	-----	0
198	VAX 11/785	UNIX 5.2	----------	-----	1.2	-----	0
199	Gould PN6005	UNIX 4.2 BSD	----------	-----	1.1	-----	0
200	VAX 11/785	UNIX 4.2 BSD	----------	-----	1.03	-----	1
201	HP 9000/500	HP-UX 4.02	B seriesCPU	-----	0.98	-----	0
202	ATT 3B20	UNIX 5.2	----------	-----	0.98	-----	0

203	VAX 11/780	UNIX 5.0.1	----------	5.0	0.93	-----	1
204	MicroVAX II	----------	----------	-----	0.92	-----	0
205	VAX 11/780	UNIX 5.2	----------	5.0	0.89	-----	0
206	Cyber DataMate	Uniplus 5.0	68010	12.5	0.89	-----	0
207	ATT PC6300+	MSDOS 3.1	80286	6.0	0.81	-----	0
208	ATT 3B2/400	UNIX 5.2	WE32100	-----	0.75	-----	0
209	Sun 2/120	Standalone	68010	10.0	0.75	-----	0
210	IBM PC/AT	VENIX/86 2.1	80286	7.5	0.75	-----	0
211	IBM PC/AT	PCDOS 3.1	80286	6.0	0.71	-----	0
212	PDP 11/70	UNIX 5.2	----------	-----	0.71	-----	0
213	Cyber DataMate	Uniplus 5.0	68010	12.5	0.71	-----	0
214	MASSCOMP 500	RTU V3.0	68010	10.0	0.70	-----	0
215	IBM PC/AT	PCDOS 3.0	80286	6.0	0.69	-----	0
216	Sun 2/120	Sun 4.2 BSD	68010	10.0	0.69	-----	0
217	ATT PC6300+	MSDOS 3.1	80286	6.0	0.69	-----	0
218	ATT PC7300	UNIX 5.2	68010	10.0	0.63	-----	0
219	VAX 11/750	VMS	----------	-----	0.62	-----	0
220	IBM PC/AT	PCDOS 3.0	80286	6.0	0.62	-----	0
221	IBM PC/AT	PCDOS 3.0	80286	6.0	0.61	-----	0
222	IBM PC/AT	VENIX/86 2.1	80286	6.0	0.57	-----	0
223	IRIS-1400	UNIX SYS V	68010	10.0	0.57	-----	0
224	Amiga 1000	AmigaDOS	68000	7.16	0.54	-----	1
225	Macintosh	Mac Rom	68000	7.8	0.52	-----	0
226	Fast Mac	----------	68000	7.7	0.51	-----	0
227	VAX 11/750	UNIX 4.2 BSD	----------	-----	0.50	-----	0
228	IBM PC/AT	PCDOS 3.0	80286	6.0	0.48	-----	0
229	ATT 3B2/300	UNIX 5.0.2	WE32000	-----	0.46	-----	0
230	ATT PC6300	MSDOS 2.11	8086	8.0	0.44	-----	0
231	NEC PC9801F	PCDOS 2.11	8086	8.0	0.44	-----	0
232	Macintosh	----------	68000	7.7	0.40	-----	0

```
233 Macintosh             Mac Rom      68000         7.8    0.40 -----   0
234 IBM PC/AT             PCDOS 3.0    80286         6.0    0.39 -----   0
235 ATT PC6300            MSDOS 2.11   8086          8.0    0.39 -----   0
236 Onyx C8002            IS/1 1.1(V7) Z8000         4.0    0.29 -----   0
237 PDP 11/34             UNIX V7M     ----------- -----    0.25 -----   0
238 IBM PC/XT             PCDOS 2.1    8088          4.77   0.23 -----   0
239 IBM PC                MSDOS 2.0    8088          4.77   0.22 -----   0
240 IBM PC                MSDOS 2.0    8088          4.77   0.19 -----   0
241 IBM PC/XT             VENIX/86 2.0 8088          4.77   0.18 -----   0
242 Cosmos                UniSoft      68000         8.0    0.18 -----   0
243 IBM PC/XT             PC/IX        8088          4.77   0.16 -----   0
244 Cromemco Z2           Cromix 11.26 Z80           4.0    0.052 ----   1
245 Apple IIe             DOS 3.3      65C02         1.02   0.021 ----   1
246 Commodore 64          C64 ROM      6510          1.0    0.0205 ---   1
```

Compiler Summary:

```
---

###

001 Compiler/Options Unknown

002 4MB cache, DEC OSF/1 AXP C X2.0, options unknown.

003 Compiler/Options Unknown

004 2MB cache, DEC OSF/1 AXP C X2.0, options unknown.

005 4MB cache, DEC OSF/1 AXP C X2.0, options unknown.

006 Compiler/Options Unknown

007 2MB cache, DEC OSF/1 AXP C X2.0, options unknown.

008 Compiler/Options Unknown

009 HP C 9.65, cc -DUNIX +O4 +Oall

010 HP C 9.61, cc -DUNIX +O4 +Oaggressive +Opipeline
    +Olibcalls -Wl,-a,archive.  New compiler improves performance
    considerably.
```

011 Compiler/Options Unknown

012 GEM C, GEM C appears to break V1.1 ...

013 cc -DUNIX

014 512KB cache, DEC OSF/1 AXP C X2.0, options unknown.

015 Compiler/Options Unknown

016 Watcom C32 V9.5 /oneatx /zp4 /5r

017 xlc 1.3.0.0, cc -DUNIX -O3

018 cc -DUNIX +P +O3 +Om1 -J -Wl,-a,archive

019 xlc V1.0.2, using Gary Karmarcik's AIX emulator (PAIX), xlc -O3 -Q

020 Dhrystone V1.1: cc -J +O3, Dhrystone V2.0: cc -J +O3 -Wl,-aarchive

021 HP92453-01 A.09.19, cc -DUNIX -J +O3 -Wl,-a,archive

022 cc 3.10.1, cc -DUNIX -O -mips2

023 HP92453-01 A.09.19, cc -DUNIX -J +O3

024 Intel Pentium Processor Performance Brief, Release 1.0, March 1993
The UNIX Compiler was: Intel Reference C Compiler (January 27th, 1993 BETA RELEASE). No other information given.

025 gcc 2.5.8, gcc -DUNIX -O2

026 HP92453-01 A.09.19, cc -DUNIX +O3

027 cc 3.10.1, cc -DUNIX -O -mips1

028 cc -DUNIX +O3

029 cc -DUNIX -O

030 HP C 9.61, cc -DUNIX +O4 +Oall

031 xlc V1.0.2, using Gary Karmarcik's AIX emulator (PAIX), xlc -O3 -Q

032 Metaware High C with '-pentium' and '-O4' options, 8 MB 64-bit RAM

033 Intel Pentium Processor Performance Brief, Release 1.0, March 1993
The UNIX Compiler was: Intel Reference C Compiler (January 27th, 1993 BETA RELEASE). No other information given.

034 HP C 9.61, cc -DUNIX +O4 +Oall

035 PowerPC 601, cc -O3 -DUNIX -DROPT

036 PowerPC 601, cc -O3 -DUNIX

037 gcc 2.5.8, gcc -O2 -DUNIX_Old -fomit-frame-pointer -finline-functions

    -fexpensive-optimizations -funroll-loops -fschedule-insns2

    -fdelayed-branch

038 XL C 3.2, cc -DUNIX -O

039 gcc 2.6.0, gcc -O2 -fexpensive-optimizations

040 gcc 2.5.4, gcc -DUNIX -DROPT -O2 -finline-functions

                    -fomit-frame-pointer

041 gcc 2.5.4, gcc -DUNIX -DROPT -O2 -finline-functions

                    -fomit-frame-pointer

042 Dhrystone V1.1: cc -J +O3, Dhrystone V2.0: cc -J +O3 -Wl,-aarchive

043 cc 3.10.1, cc -DUNIX -O2 -mips2 -sopt,-inline, 8 KByte I/D caches,

    1 MByte external (secondary) cache.

044 cc 3.10.1, cc -DUNIX -O2 -mips2 -sopt,-inline

045 Dhrystone V1.1: cc -J +O3, Dhrystone V2.0: cc -J +O3 -Wl,-aarchive

046 gcc 2.5.8, gcc -DUNIX_Old -O2 -fomit-frame-pointer -funroll-loops

                    -fexpensive-optimizations -fschedule-insns2

                    -fdelayed-branch

047 cc 3.10.1, cc -DUNIX -O2 -mips2 -sopt,-inline

048 gcc 2.5.8, gcc -DUNIX -O

049 cc 3.10.1, cc -DUNIX -O2 -mips2 -sopt,-inline, 8 KByte I/D caches,

    NO external cache. NOTE: 81.9 MIPS using cc 3.17 compilers on V1.1.

050 cc +OS +O3

051 cc -DUNIX -O -Q

052 gcc 2.4.5, gcc -DUNIX -static -O2 -mv8

053 Dhrystone V1.1: cc -J +O3, Dhrystone V2.0: cc -J +O3 -Wl,-aarchive

054 gcc 2.5.8, gcc -DUNIX_Old -O2 -fomit-frame-pointer -finline-functions

                    -fexpensive-optimizations -funroll-loops

                    -fschedule-insns2 -fdelayed-branch

055 cc -DUNIX -O

056 HP92453-01 A.09.19, cc -DUNIX

057 gcc 2.5.8, gcc -DUNIX_Old -O

058 gcc 2.4.5, gcc -DUNIX -static -O2 -mv8

059 Dhrystone V1.1: cc -J +O3, Dhrystone V2.0: cc -J +O3 -Wl,-aarchive

060 cc -DUNIX -O

061 Dhrystone V1.1: cc -J +O3, Dhrystone V2.0: cc -J +O3 -Wl,-aarchive

062 Intel Pentium Processor Performance Brief, Release 1.0, March 1993

MetaWare High C V2.4b, Compiler options not provided.

063 Sun C 2.0.1, acc -DUNIX -DROPT -Bstatic -fast -O4

The Weitek CPU runs at 2 times the SPARCstation 2 bus speed (40 MHz).

064 Sun C 2.0.1, acc -DUNIX -Bstatic -fast -O4

065 HP C 9.61, cc -DUNIX +O4 +Oall

066 HP C 9.61, cc -DUNIX +O4 +Oall

067 Borland C++ V1.0, 256K Cache, 16 MB 80 ns DRAM

068 cc -DUNIX -O

069 SC 2.0.1, acc -DUNIX -Bstatic -fast -O4

070 cc -DUNIX -DROPT -O3

071 Sun C 1.0, using cg89, cc -DUNIX -DROPT -fast -O4 -Bstatic

072 cc -O3 -DUNIX

073 cc -DUNIX -DROPT -Bstatic -O4

074 Sun C 1.0, using cg89, cc -DUNIX -fast -O4

075 Sun C 1.0, using cg89, cc -DUNIX -O

076 gcc 2.4.5, gcc -DUNIX -static -O2 -mv8 -funroll-all-loops

077 256K write-back cache, 4 MByte RAM, UMC chipset, gcc 2.2.2 (DJGPP)

gcc -DUNIX -O2 -m486 -fomit-frame-pointer -finline-functions

078 cc -DUNIX -O

079 Dhrystone V1.1: cc -J +O3, Dhrystone V2.0: cc -J +O3 -Wl,-aarchive

080 Dhrystone V1.1: cc -J +O3, Dhrystone V2.0: cc -J +O3 -Wl,-aarchive

081 gcc 2.5.7, gcc -DUNIX -O6 -m486 -funroll-loops -fomit-frame-pointer

-finline-functions

082 gcc 2.4.5, gcc -DUNIX -static -O2 -mv8

083 Metaware High C with '-486' and '-O4' options, 8 MB 64-bit RAM

084 cc -DUNIX -O -Q

085 Compiler/Options Unknown

086 gcc 2.5.8, gcc -DUNIX -m486 -O2 -fomit-frame-pointer -static
-finline-functions -funroll-loops

087 gcc 2.3.3, gcc -DUNIX -m486 -O6 -fomit-frame-pointer -Wall

088 gcc 2.3.3, gcc -DUNIX -m486 -O6 -finline-functions -funroll-all-loops

089 gcc 2.3.3, gcc -DUNIX -m486 -O2 -finline-functions

090 cc -DUNIX -O4

091 cc -DUNIX

092 gcc 2.5.6, gcc -DUNIX -static -O2

093 Watcom C32 9.5LA, Options: -oeatx -zp4, MSC timer, -oe == inline

This compiler appears to do quite well in optimizing Dhrystone.

094 cc -O2

095 Sun C 2.0.1, cc -DUNIX -DROPT -fast -O4,

Original 40 MHz motherboard replaced with 50 MHz motherboard.

096 cc 3.10.1, cc -DUNIX -O2 -sopt,-inline

097 gcc 2.4.5, gcc -DUNIX -static -O2

098 Sun C 2.0.1, acc -DUNIX -Bstatic -fast -O4

099 cc -DUNIX -O3

100 gcc 2.3.3, gcc -DUNIX -O1

101 gcc 2.3.3, gcc -DUNIX -O2

102 Intel Pentium Processor Performance Brief, Release 1.0, March 1993

MetaWare High C V2.4b. Compiler Options not provided.

103 Compiler/Options Unknown

104 cc -DUNIX -O4 -Bstatic

105 gcc 2.2.2, gcc -DUNIX -m486 -O -finline-functions -fomit-frame-pointer

106 gcc 2.2.2, gcc -DUNIX -DROPT -m486 -O2 -finline-functions
-fomit-frame-pointer

107 gcc 2.2.2, gcc -DUNIX -DROPT -m486 -O -finline-functions
-fomit-frame-pointer

108 athens.nosc.mil, /bin/cc, cc -DUNIX -O4 -Bstatic

109 Watcom C32 9.5LA, Options: -oatx -zp4, MSC timer
This compiler appears to do quite well in optimizing Dhrystone.

110 gcc 2.2.2, gcc -DUNIX -m486 -O -finline-functions
-fexpensive-optimizations

111 Sun C 1.0, cc -DUNIX -Bstatic -O4

112 acc -DCTimer -O4 -Atarg=C400

113 IBM C Set/2 V1.0, CSD Level 50, icc /G4 /O+ /Gs+ /DMSC /DROPT, 256KB
SRAM external cache, 16MB RAM.

114 cc -DUNIX -O2

115 IBM C Set/2 V1.0, CSD Level CS00042, /DMSC /DROPT /G4 /O+ /C+, 256KB
SRAM external cache, 16MB RAM.

116 cc -DUNIX -O4 -Bstatic

117 gcc 2.3.3, gcc -O9

118 cc V1.1, cc -O

119 Compiler/Options Unknown

120 COmpiler/Options Unknown

121 SAS C 6.2, global & peephole optimizers & Inline enabled.

122 cc -DUNIX -O4

123 gcc 2.5.6, gcc -DUNIX -O2

124 cc -DUNIX -O

125 Compiler/Options Unknown

126 athens.nosc.mil, /bin/cc, cc -DUNIX -O4

127 SAS C 6.2, global & peephole optimizers enabled.

128 gcc 2.3.1, gcc -O

129 cc 3.2.4, cc -DUNIX_Old -O

130 acc -DCTimer -O4 -Atarg=C100

131 cc -DUNIX -O1

132 gcc 1.93, gcc -O

133 SAS/C 6.2, global and peephole optimizers enabled.

134 gcc 2.2.2, gcc -DUNIX -DROPT -m486 -O -finline-functions

-fomit-frame-pointer

135 gcc 2.2.2, gcc -DUNIX -m486 -O -finline-functions -fomit-frame-pointer

136 gcc 2.3.3, gcc -DUNIX -m486 -O2 -finline-functions -funroll-all-loops

-fstrength-reduce

137 gcc 2.2.2, gcc -DUNIX -DROPT -m486 -O2 -finline-functions

-fomit-frame-pointer

138 gcc 2.3.3, gcc -DUNIX -O -static

139 gcc 2.3.3, gcc -DUNIX -O2

140 gcc 2.2.2, gcc -DUNIX -m486 -O2 -finline-functions -fomit-frame-pointer

141 gcc 2.2.2, gcc -DUNIX -m486 -O2 -finline-functions -funroll-loops

-fomit-frame-pointer

142 gcc 2.3.3, gcc -DUNIX -m486 -O2

143 gcc 2.4.1, gcc -DUNIX -DROPT -m486 -O -finline-functions

-fomit-frame-pointer

144 gcc 2.3.3, gcc -DUNIX -O

145 gcc 2.4.1, gcc -DUNIX -m486 -O2 -finline-functions -fomit-frame-pointer

146 gcc 2.4.5, gcc -DUNIX -O2 -fomit-frame-pointer -static

147 gcc 2.2.2, gcc -DUNIX -DROPT -m486 -finline-functions -fstrength-reduce

-fomit-frame-pointer

148 gcc 2.2.2, gcc -DUNIX -m486

149 gcc 2.3.3, gcc -DUNIX -m486 -O -finline-functions -fstrength-reduce

-fomit-frame-pointer

```
150 gcc 2.4.5, gcc -DUNIX -O6 -funroll-loops -static
151 gcc 2.2.2, gcc -DUNIX -m486 -finline-functions -O2
152 gcc 2.2.2, gcc -DUNIX -m486 -finline-functions -O
153 gcc 2.2.2, gcc -DUNIX -m486 -O
154 gcc 2.2.2, gcc -DUNIX -DROPT -m486 -O
155 gcc 2.2.2, gcc -DUNIX -DROPT -m486 -O2
156 cc 3.10.1, cc -DUNIX -O2 -sopt,-inline
157 /bin/cc, cc -O4 -dalign -Bstatic -DROPT
158 /bin/cc, cc -O4 -dalign -Bstatic
159 /bin/cc, cc -O4 -Qoption iropt -l9 -Bstatic
160 gcc 2.3.3, gcc -DUNIX -Bstatic -O
161 gcc 2.3.3, gcc -DUNIX -O
162 Compiler/Options Unknown
163 cc -DUNIX -O4 -Bstatic
164 cc 1.5, -DROPT
165 Compiler/Options Unknown
166 gcc 2.2.2, gcc -DUNIX -m486 -finline-functions -fstrength-reduce
                  -fomit-frame-pointer
167 cc -DUNIX -O
168 gcc 2.2.2, gcc -DUNIX -m486
169 gcc 2.2.2, gcc -DUNIX -DROPT -m486
170 gcc 2.2.2, gcc -DUNIX -m486
171 cc 3.2.4, cc -DUNIX_Old -O
172 gcc 2.2.2, gcc -DUNIX -DROPT
173 gcc 2.2.2, gcc -DUNIX
174 Sun C 1.0, cc -DUNIX -fast -O4 -Bstatic
175 cc -DUNIX
176 cc -DUNIX
177 gcc 1.37.1, gcc -DUNIX -O
178 cc -DUNIX
```

179 cc -DUNIX -O

180 cc -DUNIX

181 gcc 2.4.5, gcc -DUNIX -O2 -finline-functions -funroll-loops -lgcc
-DROPT -freg-struct-return

182 gcc 2.5.6, gcc -DUNIX -O2 -fomit-frame-pointer -finline-functions

183 VAX-11 C 2.0, -DROPT

184 Aztec C 5.0a, cc -dAmiga -c2 -fm -sabfmnpr
Not a great optimizing compiler ('-s' options) since a 'hand'
optimized version of the assembly code produced 7.7 V1.1 MIPS.

185 Aztec C 5.0a, cc -dAmiga -dROPT -c2 -fm -sabfmnpu

186 Aztec C 5.0a, cc -dAmiga -c2 -fm -me

187 cc V4.0, -DROPT

188 cc, -DROPT

189 gcc 2.3.3, 4 MB RAM (16-bit wide), gcc -O2 -fomit-frame-pointer
-finline-functions -m68020

190 gcc 2.3.3, 4 MB RAM (16-bit wide), gcc -O2 -fomit-frame-pointer
-finline-functions -m68020 -DROPT

191 ---

192 cc V4.0, -DROPT

193 cc, -DROPT

194 cc, -DROPT

195 cc, -DROPT

196 cc, -DROPT

197 VAX-11 C 2.0, -DROPT

198 cc, -DROPT

199 cc, -DROPT

200 cc, -DROPT

201 cc, -DROPT

202 cc, -DROPT

203 cc 4.1.1.31, -DROPT

204 cc, -DROPT

205 cc, -DROPT

206 Unisoft cc, -DROPT, used 'short' instead of 'int'.

207 CI-C86 2.20M, -DROPT

208 cc, -DROPT

209 cc, -DROPT

210 cc, -DROPT

211 Lattice C 2.15, -DROPT

212 cc, -DROPT

213 Unisoft cc, -DROPT, used 'int'.

214 cc V3.2, -DROPT

215 CI-C86 2.20M, -DROPT

216 cc, -DROPT

217 b16cc 2.0, -DROPT

218 cc, -DROPT

219 VAX-11 C 2.0, -DROPT

220 MS 3.0, small memory model

221 b16cc 2.0, -DROPT

222 cc, -DROPT

223 cc, -DROPT

224 Manx C 2.30a, -DROPT, short (16 bit) ints.

225 Mac C, -DROPT, used 'short' instead of 'int', short = 16 bits.

226 MegaMax C 2.0, -DROPT

227 cc, -DROPT

228 MS 3.0, large memory model, all other 80X8X results are small
    model under this reference.

229 cc, -DROPT

230 CI-C86 2.20M, -DROPT

231 Lattice 2.15, -DROPT

232 MegaMax C 2.0, -DROPT

```
233 Mac C, -DROPT, 32 bit int
234 CI-C86 2.1, -DROPT
235 b16cc 2.0, -DROPT
236 cc, -DROPT
237 cc, -DROPT
238 Lattice C 2.15
239 CI-C86 2.20M, -DROPT
240 b16cc 2.0, -DROPT
241 cc, -DROPT
242 cc, -DROPT
243 cc, -DROPT
244 cc, -DROPT
245 Aztec CII V1.05i, -DROPT
246 C Power 2.8, -DROPT
```

Details of who ran the benchmark:

```
---
###
REF:
0   Rick Richardson, rer@vaximile.uucp, 23 Oct 1985
    These are all early V1.0 results collected by Rick.
1   Motorola 1986 MC68020 Benchmark Report, BR353. Includes
    Table of results from Rick Richardson, PC Reasearch, Inc.
2   Al Aburto, aburto@marlin.nosc.mil, 03 Jan 1993
3   Bill Broadley, broadley@neurocog.lrdc.pitt.edu, 07 Jan 1993
4   Roger Uzun, uzun@crash.cts.com, 23 Jan 1993
5   Clark L. Coleman, clc5q@virginia.edu, 19 Feb 1993
    HP Apollo 9000 Series 700 Perfromance Brief, November 1992
6   Bob Supnik, Supnik@human.enet.dec.com, 20 Feb 1993
    Alpha AXP Server Family, Performance Brief - OpenVMS, 20 Nov 1992,
```

Second Edition, EB-N0104-51

7 Huub van Baal, huubb@neth.hp.com, 21 Feb 1993

8 Huub van Baal, huubb@neth.hp.com, 23 Feb 1993

9 Al Aburto, aburto@marlin.nosc.mil, 01 Mar 1993

10 Marc GINGOLD, marc@tobit.saclay.cea.fr, 05 Mar 1993

11 Eric S. Boltz, eboltz@nist.gov, 05 Mar 1993

12 Simon Gornall, sjg@phlim.ph.kcl.ac.uk, 09 Mar 1993

13 Andrew Mutz, mutz@kodak.com, 15 Mar 1993

14 Paul Sutcliffe Jr., paul@devon.lns.pa.us, 15 Mar 1993

15 P. Jeffrey Ungar, ungar@embezzle.stanford.edu, 16 Mar 1993

16 Fred Liu, mliu@pt.com, 20 Mar 1993

PT-SYS5K is a SPARC system designed for real time application.

17 Al Aburto, a@cs.umr.edu, 22 Mar 1993

19 Mario A. Guerra, mguerra@inforisc.cr, 01 Apr 1993

20 Benno Senoner, senoner@ghost.sm.dsi.unimi.it, 02 Apr 1993

21 Peter Valkenburg, valke@cca.vu.nl, 15 Apr 1993

22 Maurice Janssen, mhmj@chem.vu.nl, 16 Apr 1993

23 Benno Senoner, senoner@ghost.sm.dsi.unimi.it, 28 Apr 1993

24 Piero Serini, piero@strider.st.dsi.unimi.it, 1virginia.edu, 23 Jul 1993

28 John E. Stone, johns@cs.umr.edu, 27 Jul 1993

29 Roger Uzun, uzun@crash.cts.com, 19 Aug 1993

30 Al Aburto, aburto@marlin.nosc.mil, 06 Sep 1993

31 Michael Chapman, mike@hopper.acs.virginia.edu, 15 Oct 1993ar,

bhandarkar@msbcs.enet.dec.com, 15 Oct 1993

Posted to comp.benchmarks.

34 Henrik Harmsen, d9hh@dtek.chalmers.se, 16 Oct 1993

35 Skip Sauls, skip@cy.cs.olemiss.edu, 18 Oct 1993

36 18 Oct 1993

37 Lionel Lacassagne, lacass_l@boson.epita.fr, 23 Oct 1993

38 Roger Uzun, uzun@crash.cts.com, 31 Oct 1993

39 Ronald Schalk, R.Schalk@uci.kun.nl, 04 Nov 1993

40 Michael Ritzert, mjr@dyn.geo.uni-bonn.de, 13 Nov 1993

41 Al Aburto, aburto@ariel.nosc.mil, 28 Nov 1993

42 Al Aburto, aburto@octopus.nosc.mil, 12 Dec 1993

43 Kristian Wedberg, wedberg@mednet.gu.se, 16 Dec 1993

44 Petri Nordlund, petrin@mits.mdata.fi, 18 Dec 1993

45 Al Aburto, aburto@marlin.nosc.mil, 08 Jan 1994

46 Bo Thide', bt@irfu.se, 17 Jan 1994

47 Pekka J{rvel{inen, jarvelai@csc.fi, 15 Feb 1994

48 nsa@link.hacktic.nl, 23 Feb 1994

49 Wolfram Wagner, ww@mpi-sb.mpg.de, 10 Mar 1994

50 Wolfram Wagner, ww@mpi-sb.mpg.de, 28 Apr 1994

51 Dennis J Robinson, cs370_4@eecs.uic.edu, 06 May 1994

52 Bill Broadley, broadley@neurocog.lrdc.pitt.edu, 08 May 1994

53 Harlan W Stockman, hwstock@saix531.energylan.sandia.gov, 18 May 1994

54 Georg Wambach, gw@informatik.uni-koeln.de, 25 May 1994

55 Bor-Wen Jeng, bwjeng@beta.wsl.sinica.edu.tw, 16 Jun 1994

56 Evan Torrie, torrie@cs.stanford.edu, 21 Jun 1994

57 Pawel Potocki, ppotocki@panix.com, 29 Jun 1994

58 Wolfram Wagner, ww@mpi-sb.mpg.de, 19 Jul 1994

59 Mario Guerra, mguerra@cariari.ucr.ac.cr, 10 Aug 1994

60 DANI, dani@morgana.uab.es, 23 Aug 1994

61 Zygmunt Krawczyk, zkr@ita.pwr.wroc.pl, 09 Sep 1994

62 Al Aburto, aburto@marlin.nosc.mil, 05 Nov 1994

Al Aburto

aburto@marlin.nosc.mil

INDEX

Revision History:

Rev 1.0, 4/1/95, DHG, Initial writing
Rev 1.1, 5/1/95, DHG, found correct formula = template for user benchmarks
Rev 1.2, 5/5/95, DHG, error correction
Rev 1.3, 5/8/95, DHG, added more locate info
Rev 1.4, 5/9/95, DHG, format changes, made internet stuff and appendix
Rev 1.5, 5/10/95, DHG, added screens for PDRT186, clean up
Rev 1.6, 5/12/95, DHG, added screens for make/compile/assemble/locate
Rev 1.7, 5/17/95, DHG, added quick start
Rev 1.8, 5/18/95, DHG, AMD formating applied
Rev 1.9, 5/19/95, DHG, clean-up, error correction
Rev 2.0, 5/20/95, DHG, addition of problem reports from alpha sites, index
Release to EPD for final editing.