1785

The University of Georgia

4235 Group 7: Library Components 1

Connlaoi Fruit, Daniel Covaci

cwf94894@uga.edu, dc53339@uga.edu

3/14/2024

E4235_Delaycenti(int cs_count)

Definitions

At time of writing, there are no defines necessary for the function, however this may change once functionality is combined with the getCPU function to allow for multiple CPU speeds to use the functions.

Final Inputs

R0 = The number centiseconds to delay for.

Final Outputs

N/A

Function Use

To delay the continuation of a program by a number of centiseconds. The function assumes a positive decimal input of type int. This limits the range of centiseconds able to be delayed via the function as between 1 to 2,147,483,647. At time of writing, the function assumes a clock speed of 1.8 GHz to function properly.

Example ASM

```
CS:
   ldr r0, =6000
   bl E4235_Delaycenti
   b exit
```

Example C

```
//
// C program to call Assembly delays
//

#include <stdio.h>
extern void E4235_Delaynano(int);
extern void E4235_Delaymicro(int);
extern void E4235_Delaymilli(int);
extern void E4235_Delaycenti(int);
extern void E4235_Delaysec(int);
extern void E4235_Delaymin(int);
```

```
int main()
{
    int ns = 500000000;   // Nanoseconds/sec
    int mcs = 60000000;       // Microseconds/min
    int mis = 60000;    // Milliseconds/min
    int cs = 6000;             // Centiseconds/min
    int sec = 60;                     // Seconds/min
    int min = 1;                      // Minutes/min

    // E4235_Delaynano(ns);    // CHECKED
    // E4235_Delaymicro(mcs);   // CHECKED
    // E4235_Delaymilli(mis);    // CHECKED
    E4235_Delaycenti(cs);    // CHECKED
    // E4235_Delaysec(sec);   // CHECKED
    // E4235_Delaymin(min);    // CHECKED

    return(0);
}
```

Test Results

We tested the code by running each test program and simultaneously timing the stopwatch on our phones. After the end of the program ends, we stop the stopwatch and compare the accuracy of the value from the stopwatch to the input delay. For a set delay of 1 min, we clocked a delay of 1.04 min

E4235_Delaynano(int ns_count)

Definitions

At time of writing, there are no defines necessary for the function, however this may change once functionality is combined with the getCPU function to allow for multiple CPU speeds to use the functions.

Final Inputs

R0 = The number of nanoseconds to delay for

Final Outputs

N/A

Function Use

To delay the continuation of a program by a number of nanoseconds. The function assumes a positive decimal input of type int. This limits the range of nanoseconds able to be delayed via the function as between 1 to 2,147,483,647. At time of writing, the function assumes a clock speed of 1.8 GHz to function properly. The E4235_Delaynano function will be increasingly inaccurate for all CPU speeds below 2 GHz due to the hardware limitations of such a small increment of time needing to be delayed in a function of this form. As previously noted, all testing was performed with a 1.8 GHz CPU and this affected our testing results.

Example ASM

```
NS:
   ldr r0, = 500000000 @ 500,000,000 ns delay for 1/2 sec
   bl E4235_Delaynano
   b exit
```

Example C

```
//
// C program to call Assembly delays
//

#include <stdio.h>
extern void E4235_Delaynano(int);
extern void E4235_Delaymicro(int);
extern void E4235_Delaymilli(int);
extern void E4235_Delaycenti(int);
extern void E4235_Delaysec(int);
extern void E4235_Delaymin(int);

int main()
{
   int ns = 500000000;    // Nanoseconds/sec
   int mcs = 60000000;           // Microseconds/min
   int mis = 60000;       // Milliseconds/min
   int cs = 6000;                     // Centiseconds/min
   int sec = 60;                                  // Seconds/min
   int min = 1;                                   // Minutes/min

   E4235_Delaynano(ns);     // CHECKED
   // E4235_Delaymicro(mcs);    // CHECKED
   // E4235_Delaymilli(mis);     // CHECKED
   // E4235_Delaycenti(cs);     // CHECKED
   // E4235_Delaysec(sec);    // CHECKED
   // E4235_Delaymin(min);     // CHECKED

   return(0);
}
```

Test Results

We tested the code by running each test program and simultaneously timing the stopwatch on our phones. After the end of the program ends, we stop the stopwatch and compare the accuracy of the value from the stopwatch to the input delay. For a set delay of 2 seconds, we clocked a delay of 2.9 seconds. In addition to this, we utilized the code from the BlinkC assignment along with our delay function to generate a square wave signal at a set delay for each. We used a half-second delay ideally generating a 1 Hz signal, which was measured on an oscilloscope at .9 Hz.

E4235_Delaymicro(int mcs_count)

Definitions

At time of writing, there are no defines necessary for the function, however this may change once functionality is combined with the getCPU function to allow for multiple CPU speeds to use the functions.

Final Inputs

R0 = The number of microseconds to delay for

Final Outputs

N/A

Function Use

To delay the continuation of a program by a number of microseconds. The function assumes a positive decimal input of type int. This limits the range of microseconds able to be delayed via the function as between 1 to 2,147,483,647. At time of writing, the function assumes a clock speed of 1.8 GHz to function properly.

Example ASM

```
McS:
   ldr r0, =60000000
   bl E4235_Delaymicro
   b exit
```

Example C

```
//
// C program to call Assembly delays
//

#include <stdio.h>
extern void E4235_Delaynano(int);
extern void E4235_Delaymicro(int);
extern void E4235_Delaymilli(int);
extern void E4235_Delaycenti(int);
extern void E4235_Delaysec(int);
extern void E4235_Delaymin(int);
```

```
int main()
{
    int ns = 500000000;    // Nanoseconds/sec
    int mcs = 60000000;            // Microseconds/min
    int mis = 60000;       // Milliseconds/min
    int cs = 6000;                 // Centiseconds/min
    int sec = 60;                           // Seconds/min
    int min = 1;                            // Minutes/min

    // E4235_Delaynano(ns);     // CHECKED
    E4235_Delaymicro(mcs);    // CHECKED
    // E4235_Delaymilli(mis);     // CHECKED
    // E4235_Delaycenti(cs);     // CHECKED
    // E4235_Delaysec(sec);    // CHECKED
    // E4235_Delaymin(min);     // CHECKED

    return(0);
}
```

Test Results

We tested the code by running each test program and simultaneously timing the stopwatch on our phones. After the end of the program ends, we stop the stopwatch and compare the accuracy of the value from the stopwatch to the input delay. For a set delay of 1 min, we clocked a delay of 1.02 min

E4235_Delaymilli(int mis_count)

Definitions

At time of writing, there are no defines necessary for the function, however this may change once functionality is combined with the getCPU function to allow for multiple CPU speeds to use the functions.

Final Inputs

R0 = The number of milliseconds to delay for

Final Outputs

N/A

Function Use

To delay the continuation of a program by a number of milliseconds. The function assumes a positive decimal input of type int. This limits the range of milliseconds able to be delayed via the function as between 1 to 2,147,483,647. At time of writing, the function assumes a clock speed of 1.8 GHz to function properly.

Example ASM

```
MiS:
   ldr r0, =60000
   bl E4235_Delaymilli
   b exit
```

Example C

```
//
// C program to call Assembly delays
//

#include <stdio.h>
extern void E4235_Delaynano(int);
extern void E4235_Delaymicro(int);
extern void E4235_Delaymilli(int);
extern void E4235_Delaycenti(int);
extern void E4235_Delaysec(int);
extern void E4235_Delaymin(int);
```

```
int main()
{
   int ns = 500000000;    // Nanoseconds/sec
   int mcs = 60000000;          // Microseconds/min
   int mis = 60000;       // Milliseconds/min
   int cs = 6000;               // Centiseconds/min
   int sec = 60;                          // Seconds/min
   int min = 1;                           // Minutes/min

   // E4235_Delaynano(ns);     // CHECKED
   // E4235_Delaymicro(mcs);    // CHECKED
   E4235_Delaymilli(mis);     // CHECKED
   // E4235_Delaycenti(cs);     // CHECKED
   // E4235_Delaysec(sec);    // CHECKED
   // E4235_Delaymin(min);     // CHECKED

   return(0);
}
```

Test Results

We tested the code by running each test program and simultaneously timing the stopwatch on our phones. After the end of the program ends, we stop the stopwatch and compare the accuracy of the value from the stopwatch to the input delay. For a set delay of 1 min, we clocked a delay of 1.04 min

E4235_Delaysec(int sec_count)

Definitions

At time of writing, there are no defines necessary for the function, however this may change once functionality is combined with the getCPU function to allow for multiple CPU speeds to use the functions.

Final Inputs

R0 = The number of seconds to delay for

Final Outputs

N/A

Function Use

To delay the continuation of a program by a number of seconds. The function assumes a positive decimal input of type int. This limits the range of seconds able to be delayed via the function as between 1 to 2,147,483,647. At time of writing, the function assumes a clock speed of 1.8 GHz to function properly.

Example ASM

```
Sec:
   ldr r0, =60
   bl E4235_Delaysec
   b exit
```

Example C

```
//
// C program to call Assembly delays
//

#include <stdio.h>
extern void E4235_Delaynano(int);
extern void E4235_Delaymicro(int);
extern void E4235_Delaymilli(int);
extern void E4235_Delaycenti(int);
extern void E4235_Delaysec(int);
extern void E4235_Delaymin(int);
```

```
int main()
{
    int ns = 500000000;    // Nanoseconds/sec
    int mcs = 60000000;            // Microseconds/min
    int mis = 60000;       // Milliseconds/min
    int cs = 6000;                 // Centiseconds/min
    int sec = 60;                          // Seconds/min
    int min = 1;                           // Minutes/min

    // E4235_Delaynano(ns);     // CHECKED
    // E4235_Delaymicro(mcs);    // CHECKED
    // E4235_Delaymilli(mis);     // CHECKED
    // E4235_Delaycenti(cs);     // CHECKED
    E4235_Delaysec(sec);    // CHECKED
    // E4235_Delaymin(min);     // CHECKED

    return(0);
}
```

Test Results

We tested the code by running each test program and simultaneously timing the stopwatch on our phones. After the end of the program ends, we stop the stopwatch and compare the accuracy of the value from the stopwatch to the input delay. For a set delay of 1 min, we clocked a delay of 1.06 min

E4235_Delaymin(int min_count)

Definitions

At time of writing, there are no defines necessary for the function, however this may change once functionality is combined with the getCPU function to allow for multiple CPU speeds to use the functions.

Final Inputs

R0 = The number of minutes to delay for

Final Outputs

N/A

Function Use

To delay the continuation of a program by a number of minutes. The function assumes a positive decimal input of type int. This limits the range of minutes able to be delayed via the function as between 1 to 2,147,483,647. At time of writing, the function assumes a clock speed of 1.8 GHz.

Example ASM

```
Min:
   ldr r0, =1
   bl E4235_Delaymin
   b exit
```

Example C

```
//
// C program to call Assembly delays
//

#include <stdio.h>
extern void E4235_Delaynano(int);
extern void E4235_Delaymicro(int);
extern void E4235_Delaymilli(int);
extern void E4235_Delaycenti(int);
extern void E4235_Delaysec(int);
extern void E4235_Delaymin(int);
```

```
int main()
{
    int ns = 500000000;    // Nanoseconds/sec
    int mcs = 60000000;            // Microseconds/min
    int mis = 60000;       // Milliseconds/min
    int cs = 6000;                      // Centiseconds/min
    int sec = 60;                                  // Seconds/min
    int min = 1;                                   // Minutes/min

    // E4235_Delaynano(ns);      // CHECKED
    // E4235_Delaymicro(mcs);    // CHECKED
    // E4235_Delaymilli(mis);     // CHECKED
    // E4235_Delaycenti(cs);     // CHECKED
    // E4235_Delaysec(sec);    // CHECKED
    E4235_Delaymin(min);     // CHECKED

    return(0);
}
```

Test Results

We tested the code by running each test program and simultaneously timing the stopwatch on our phones. After the end of the program ends, we stop the stopwatch and compare the accuracy of the value from the stopwatch to the input delay. For a set delay of 1 min, we clocked a delay of 1.03 min