

4235 Group 7: Library Components 1

Connlaoi Fruit, Daniel Covaci

[cwf94894@uga.edu](mailto:cwf94894@uga.edu), [dc53339@uga.edu](mailto:dc53339@uga.edu)

3/14/2024

# 

[**E4235\_Delaycenti(int cs\_count) 4**](#_rx7c4fbzndss)

[Definitions 4](#_z9jyazr2t0gt)

[Final Inputs 4](#_5t6w9s4d8gef)

[Final Outputs 4](#_e5v0xsz7u1iu)

[Function Use 4](#_dohdovaa4bam)

[Example ASM 4](#_mt1609ikg7nz)

[Example C 4](#_enemu3yx344w)

[Test Results 5](#_6a6y2kj5b613)

[**E4235\_Delaynano(int ns\_count) 6**](#_rsma7i3y1gb9)

[Definitions 6](#_1enzulde73np)

[Final Inputs 6](#_wlex72jtf8do)

[Final Outputs 6](#_j1h04k7wm99n)

[Function Use 6](#_kcatep6fzb0c)

[Example ASM 6](#_vtkq1nwrofx0)

[Example C 6](#_xizh72xqjzn7)

[Test Results 7](#_vcf432r9l1mj)

[**E4235\_Delaymicro(int mcs\_count) 8**](#_bpbgrm9ec2tg)

[Definitions 8](#_wqzb1isvjbcw)

[Final Inputs 8](#_j6ggx6q1oq3l)

[Final Outputs 8](#_sd3euo7nr60r)

[Function Use 8](#_bi5k65am1kxr)

[Example ASM 8](#_1quo9ceeutkm)

[Example C 8](#_7n58zfmy6yky)

[Test Results 9](#_c84iiypekocf)

[**E4235\_Delaymilli(int mis\_count) 10**](#_renq9m80m002)

[Definitions 10](#_txphl6j230f8)

[Final Inputs 10](#_r4r8gobzwpx7)

[Final Outputs 10](#_8o66sh1agzee)

[Function Use 10](#_cqsokn64q4cm)

[Example ASM 10](#_cauckon0fr8h)

[Example C 10](#_bgyqh72fvce3)

[Test Results 11](#_qygduyexcdnv)

[**E4235\_Delaysec(int sec\_count) 12**](#_u8jzqjqwv1ds)

[Definitions 12](#_vusvpdbm0x67)

[Final Inputs 12](#_qa5239607vgq)

[Final Outputs 12](#_p6ft2rrrieom)

[Function Use 12](#_4j2rxglcx0c4)

[Example ASM 12](#_muupe528o8bx)

[Example C 12](#_nuwg73pi56sn)

[Test Results 13](#_3ip9zn4vtqz7)

[**E4235\_Delaymin(int min\_count) 14**](#_j72lgk9yd4r4)

[Definitions 14](#_ag2qvgxrtvcg)

[Final Inputs 14](#_qqtfvzq99dae)

[Final Outputs 14](#_gpdettt6oicw)

[Function Use 14](#_lgroohy07wo1)

[Example ASM 14](#_493najrrchnr)

[Example C 14](#_h8ql5knnxf2x)

[Test Results 15](#_z0g0sxj5kzno)

# 

# 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