**Library 2**

ECSE4235: Embedded Systems II
Small Project – 2024 – Timothy Million, Sreya Bitra


UNIVERSITY OF
GEORGIA

# Documentation

## MultiRead

The multiread function is designed to read multiple GPIO values at the same time, mimicking a bus.

uint16_t E4235_multiread (int [], int)

**Parameters:**

r0 → array of pin numbers

- If a series of numerically ordered pins are wanted, the format [#, '-', #] can be used inside the array
- Cannot exceed 16 pins

r1 → length of the array

**Returns:**

The values of the multiple GPIOs being read. -1 is returned if invalid

## MultiWrite

The multiwrite function is used to be able to write to multiple GPIO pins at the same time, mimicking a bus.

int E4235_multiwrite (int[], int, uint16_t)

**Parameters:**

r0 → array of pin numbers

- If a series of numerically ordered pins are wanted, the format [#, '-', #] can be used
- Cannot exceed 16 pins
- If a pin is repeated the value is assigned to the second time will be written

r1 → length of the array

r2 → value to write

**Returns:**

-1 if invalid

# Appendix

## E4235_multiwrite

```
        .global E4235_multiwrite
        .extern E4235_Write

E4235_multiwrite:
                push {r3 - r12, lr}     @ r0 = user gpio array, r1 = length of array, r2 = value to write
                ldr r10, =value
                str r2, [r10]           @ store value to write in value variable
                mov r7, #0                      @ r7 holds number of gpio pins entered
                ldr r3, =arr            @ arr holds the actual gpio pin numbers
iterate:
                ldr r4, [r0], #4
                cmp r4, #45                     @ decimal equivalent of '-' character
                bne store                       @ if not in "# - #" format, store normally
                sub r1, r1, #1
                ldr r5, [r0, #-8]       @ r5 holds starting value in "# - #" format
                ldr r6, [r0]            @ r6 holds ending value in "# - #" format
inbtw:                                  @ iterates through "# - #" notation
                add r5, r5, #1
                str r5, [r3], #4
                add r7, r7, #1
                cmp r5, r6
                bne inbtw
                add r0, r0, #4
                b count
store:
                str r4, [r3], #4
                add r7, r7, #1
count:
                sub r1, r1, #1
                cmp r1, #0
                bne iterate
                mov r5, #-1                     @ ending character of arr
                str r5, [r3]
                cmp r7, #16                     @ maximum amount of pins
                bgt errorpin
                ldr r4, =arr
                ldr r5, =value
write:
                @ insert custom binary vector parsing
                ldr r6, [r5]            @ r6 holds the value to be set
                cmp r7, #0
                beq end
                                                        @ value parsing
                mov r0, #16                     @ max amount of pins
                sub r8, r0, r7
                add r8, r8, r0          @ amount to lsl by
                lsl r6, r6, r8          @ get rid of prior bits
                lsr r6, #31                     @ get rid of later bits
                                                        @ r6 holds level of certain bit
                ldr r0, [r4], #4        @ r0 = pin to set level
                mov r1, r6                      @ r1 = level value
                bl E4235_Write
                cmp r0, #-1                     @ if error
                beq end                         @ end function and return -1
                sub r7, r7, #1
                b write
end:
                pop {r3 - r12, lr}
                bx lr
errorpin:
                ldr r0, =pinerrormsg
                bl printf
                mov r0, #-1
                b end

                .data
                pinerrormsg: .ascii "The number of pins is not valid.\n"
                arr: .zero 131072
                value: .zero 64
```

## E4235_multiread

```
        .global E4235_multiread
        .extern E4235_read

E4235_multiread:
                push {r2 - r12, lr}     @ r0 = user gpio array, r1 = length of array
                ldr r3, =arr
iterate:
                ldr r4, [r0], #4
                cmp r4, #45                     @ decimal for '-' character
                bne store                       @ if not in "# - #" format, store normally
                sub r1, r1, #1
                ldr r5, [r0, #-8]       @ r5 holds starting value in "# - #" format
                ldr r6, [r0]            @ r6 holds ending value in "# - #" format
inbtw:                                          @ iterates through "# - #" notation
                add r5, r5, #1
                str r5, [r3], #4
                cmp r5, r6
                bne inbtw
                add r0, r0, #4
                b count
store:
                str r4, [r3], #4
count:
                sub r1, r1, #1
                cmp r1, #0
                bne iterate
                mov r5, #-1                     @ ending character of arr
                str r5, [r3]
                cmp r7, #16                     @ maximum amount of pins
                bgt errorpin
                ldr r4, =arr
                ldr r5, =in
read:
                ldr r0, [r4], #4
                cmp r0, #-1                     @ looking for ending character
                beq endok
                bl E4235_read           @ input(r0 before) = gpio pin number, output(r0 after) = level value
                ldr r1, =in
                ldr r1, [r1]
                lsl r1, #1
                add r0, r1, r0
                str r0, [r5]
                b read
endok:
                ldr r0, =in
                ldr r0, [r0]
end:
                pop {r2 - r12, lr}
                bx lr
errorpin:
                ldr r0, =pinerrormsg
                bl printf
                mov r0, #-1
                b end

                .data
                pinerrormsg: .ascii "The number of pins is not valid.\n"
                in: .zero 16
                arr: .zero 131072
```

**multirw_ctest.c**

```c
#include <stdio.h>
#include <stdint.h>

//multigpio(int array, int length, int value);
extern uint16_t E4235_multiread(int[], int);
extern void E4235_multiwrite(int[], int, uint16_t);

int arrin[] = {22, 6};
int arrout[] = {23, '-', 25, 26};

int main() {

  E4235_multiwrite(arrin, 2, 0x0002);
  uint16_t bin = E4235_multiread(arrout, 4);

  printf("%d\n", bin);
  return (0);
}
```

**multirw_asmtest.s**

```
            .text
            .global main

 main:
            ldr r0, =arrin
            mov r1, #2
            mov r2, #2
            bl E4235_multiwrite

            ldr r0, =arrout
            mov r1, #4
            bl E4235_multiread

            bl printf

            .data
            arrin: .word 22, 6
            arrout: .word 23, '-', 25, 26
```

**Expected output of tests:** depending on how the pins are connected on the breadboard, the pins connected to GPIO_22 should read 1, and the pins connected toGPIO_ 6 should read 0

**References**

1. “BCM2711." Raspberry Pi Documentation, Raspberry Pi Foundation, https://www.raspberrypi.com/documentation/computers/processors.html#bcm2711.