Timer Using Loop Delays

Course: ENGG*3640 Microcomputer Interfacing

Instructor: Radu Muresan

Student Names/Numbers:

Ali Akhdar (1068542)

Bilal Ayyache (0988616)

Mohammed Al-Fakhri (0982745)

Emeka Madubuike (0948959)

Date: 26/09/2018

Contents

1. Introduction

The Purpose of lab 1 was to get familiar with the Kiel Uvision program interface and learn the basics of collaborative programming between C language and assembly.

1.1. Lab Description

The first part of the lab was focused on setting up the Kiel Uvision program and create a template that would as a workstation. The second task was focused on creating a short program that loads two values into registers from which a C program calls the assembly code to make a copy of the src string and print the results to the putty screen. The third task of the first lab was to design and implement a delay loop. The delay loop acts as a timer that prints the count every ten seconds.

1.2. System Requirements

During the first lab, the tools and equipment that was used were the following:

- Kiel Uvision Program was used to create instructions to the K60 Microcontroller
- FreeScale TWR-K60D100M Microcontroller was used to implement instructions sent
- Connection cable between PC and board was used to connect the Microcontroller to the PC

2. Background

2.1. EQUIPMENTS

- **Kiel Uvision Program**: The Kiel Uvision program is an IDE that combines project management, source code editing, program debugging, and run-time environment into a single powerful environment. Using this environment, the user is able to easily and efficiently test, verify, debug, and optimize the code developed. During the first lab, the debug functionality helped in understanding how the code is acting.
- **K60 Microcontroller**: The K60 Microcontroller (Figure 2.1) from NXP contains a low power MCU core ARM Cortex-M4 that features an analog integration, serial communication, USB 2.0 full-speed OTG controller and 10/100 Mbps Ethernet MAC. These characteristics makes this Microcontroller suitable to preform task in a very efficient and fast manner. A USB connection was used to sync the microcontrollerwith the Keil uVision software that that was provided by the Teacher Assistant.

Figure 2.1: K60 Microcontroller

3. Implementation

3.1. Lab Implementation Overview

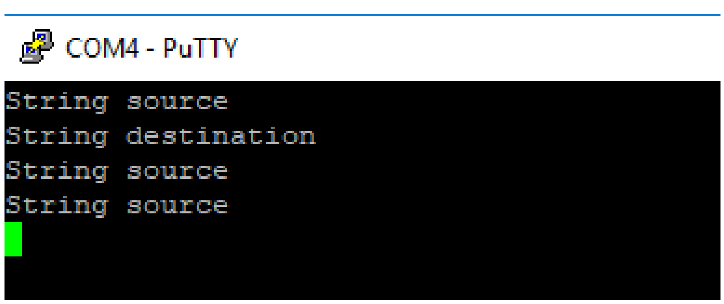


Figure 3.1.1: String printed Properly after troubleshooting

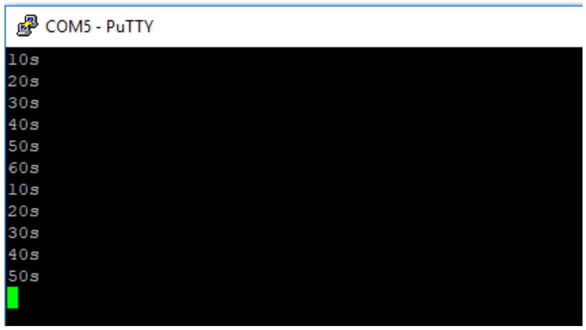


Figure 3.1.2: Counter and Loop Delay output

3.1.1. Software Implementation

The main.c that is used in part one (Figure A.1: Page 1 main.c) of the lab was provided in the lab manual. Simply, the main.c has the definitions and functions that are used in assembly and that are extremely essential to the workings of microcontroller. Lines 35-39 contains the standard C included files, The printing function which displays the output on putty are shown in Lines 42-45. From line 46-51 is the entry function that is needed only for the entry point into the assembly code mymain.s.

The mymain.s (Figure B.1: Page 1 myMain.s) in part one was written using ARM assembly language, Lines 6-11 contains the instructions needed to set the string in the registers and printing it. Lines 14-21, loads the bytes from source(r1) into the carrier(r3) and updates the address, then stores it from the carrier to the destination (r0) and updates the address. It then it checks for a zero terminator (last byte), loads the registers with strings and prints them. Lines 24-28 comprises simple instructions to assign the strings that had to be stored in the registers and displayed on putty.

For part two, the mymain.s (Figure C.1: Page 1 myMain.s) contains the implementation of the delay program that is required. As it is shown in Figure C.1, after zero is loaded into register r5, the “tim” loop, which is the main loop that the program runs on, begins by loading r1 with a number large enough to aid in delaying the delay timer to approximately 10 seconds. The number used is 0xF2EDE0(15920608) and it was procured after some calculations and an extensive period of trial and error that are better explained in the below section Lab 2 Requirements. Then a 4 instruction delay loop is created “Dloop” between line 15-18 having subtraction by 1 from register r1 and comparing it by zero and a NOP instruction. As we go outside this delay loop we start adding one to r5 register (Line 20) as the “tim” loop is still running again and again it will continue adding r5 by one till it reaches any of the 10,20,30,40,50 or 60. Every time we increment r5 we check if one of these numbers is reached as shown in lines 23-34, if any of these numbers hits the, code branches to load r0 (Line 39-63) and then prompts the correct string that are store between line 65-72, and then “tim” loop starts the same procedure again.

3.1.2. Hardware Implementation

During the first lab, Implementation of hardware was minimal. The TWR-K60D100M Microcomputer was used to demonstrate the code implementation. Putty terminal was used to display the results on screen. Throughout lab 1, it was noticed that the on-board crystal frequency was valued at 50 MHz. The above frequency value was used to calculate the register preload values.

3.2. Lab Requirements

3.2.1 Lab 1 Requirement

The mymain.s program that was provided from the lab manual contained an intentional code error. The second task of lab 1 was to debug the program and fix the error to print the expected result. After carefully debugging the provided program, it was observed that register 0 gets cleared every time the “myprint” function was called. To solve this error, register 1 had to be reloaded with an immediate offset. Register 1 had to be reloaded with the previous string in order for the compare function to work. The following code line “LDR r0 = dststr” was included before r1 was loaded with the source string.

Through monitoring the disassembly window and using the debugger mode, it was discovered that the “dststr” and the “srcstr” were both located in the data memory at location 0x1FFF002E. It was also found that the Main.c file was located at 0x00023C0 and assmain at 0x0000083C. The screenshots of these locations can be seen below:

```
0x000023C6 F7FEFA39   BL.W      asmmain (0x0000083C)
```

Figure 3.2.1.1

Call Stack + Locals

Name	Location/V...	Type
main	0x000023C0	int f()

Figure 3.2.1.2

Core	
R0	0x1FFF002E
R1	0x1FFF002E

Figure 3.2.1.3

3.2.2 Lab 2 Requirement

As for this program, it’s required to develop program that counts by 10 seconds and displays the seconds repeatedly within approximately 10 second apart. To achieve this, a delay loop was implemented with 4 lines of code shown in the figure above including one NOP (No Operation Instruction). The TWR-K60D100M has an innate crystal speed of 50 MHz, this was used for getting the processor cycles as shown below:

$$T = \frac{1}{50Mhz}$$
$$T = 2\,x\,10^{-8}s\ (per\ cycle)$$
$$\frac{1s}{2\,x\,10^{-8}} = 50{,}000{,}000\ instructions\ a\ second$$

Figure 3.2.2: Formula used to figure out time per cycle

This number is divided by 4 because of the 4 instructions that we used in our delay loop, then converted to hexadecimal and loaded to r1 (loop register). After each loop exit, the r0 register which is set to 0 initially will increment by 1. The result of r0 in each iteration is compared to 10,20,30,40,50,60 and every time it hits a number within these, it prompts it on the screen. By convention, if r0 is incrementing from 0 it will hit 10 first and finishes with 60 thus displaying the result in a specific sequence and time this code is shown above.

3.3. Simulation Results

3.3.1. Simulation Result Part 1

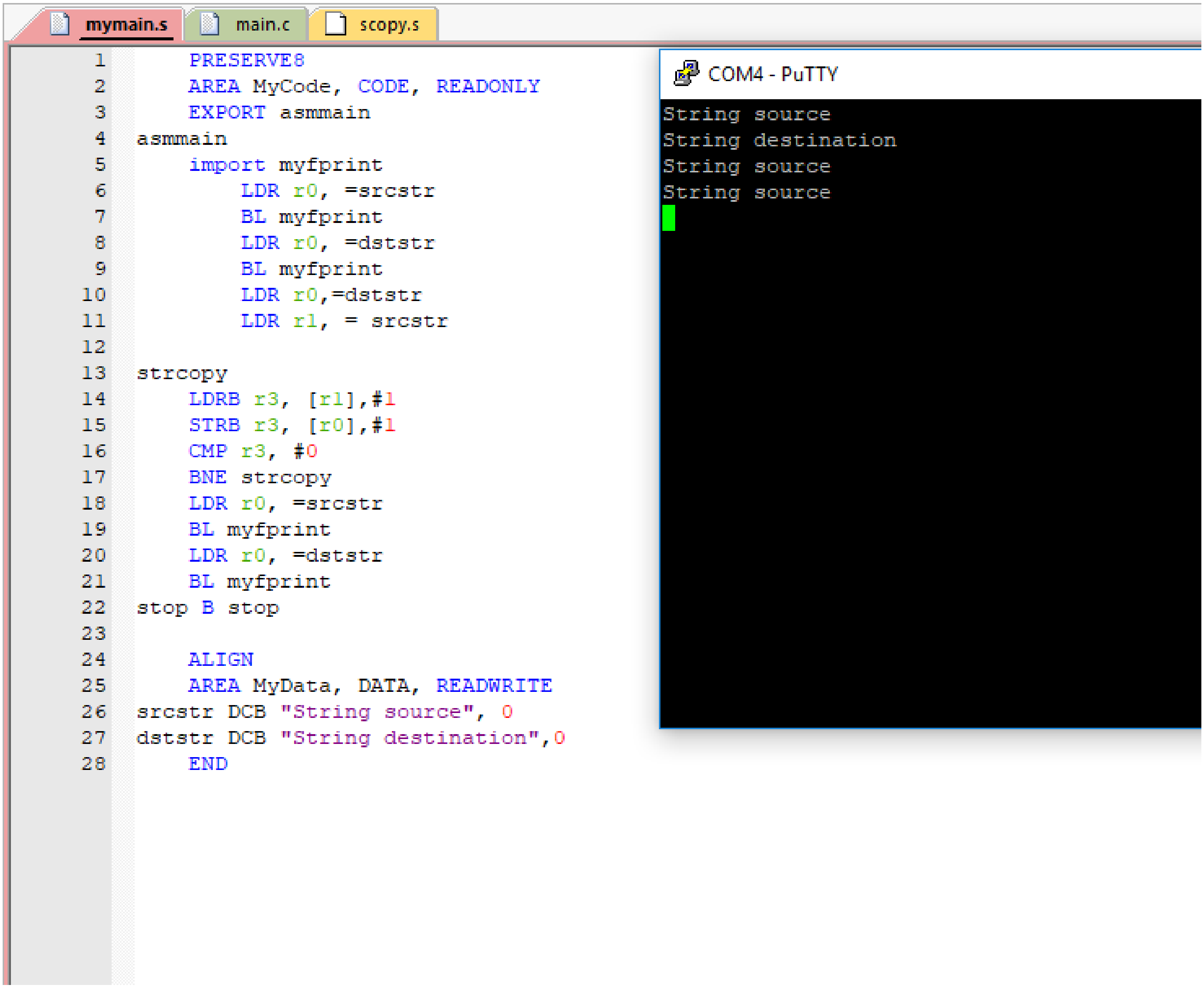
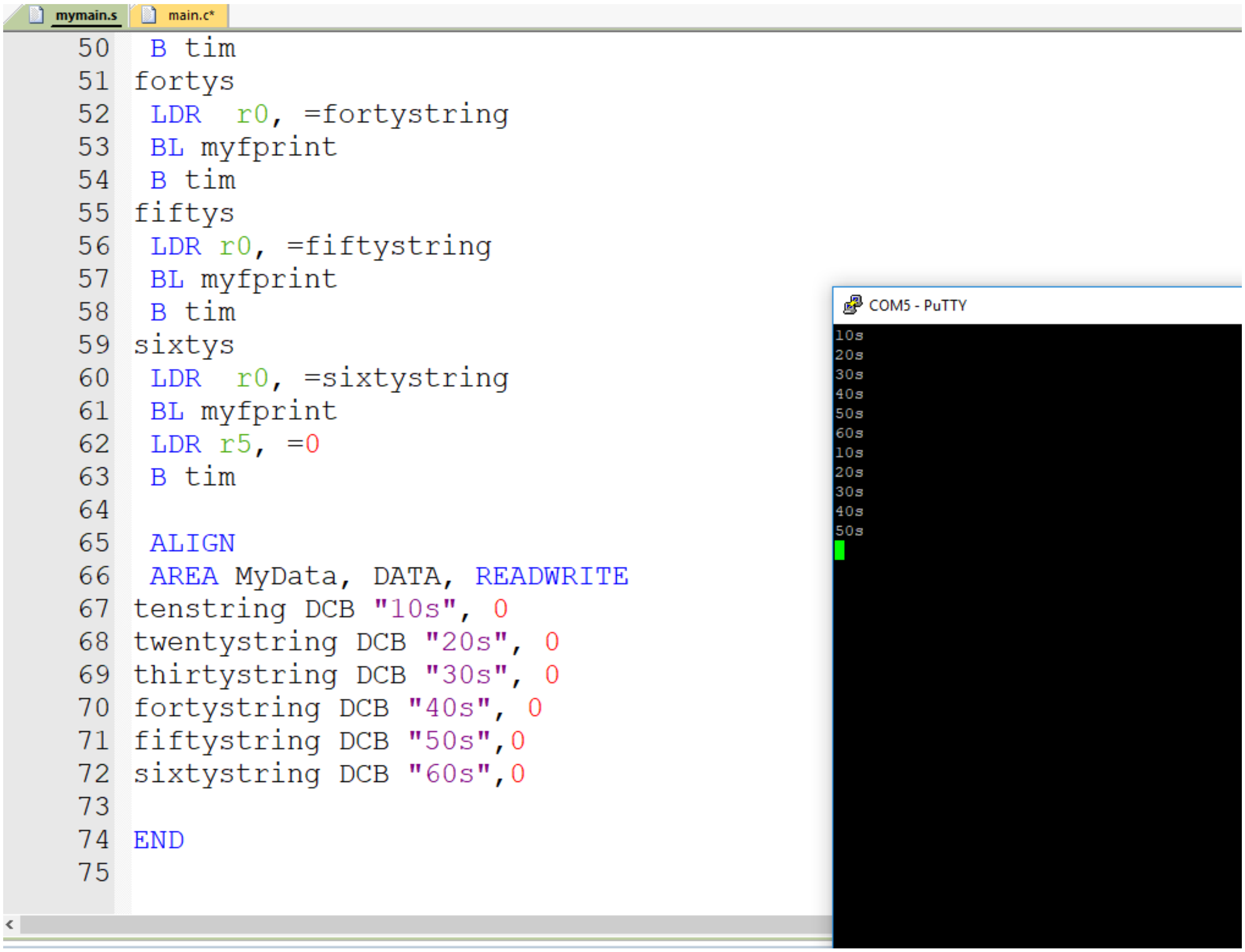


Figure 3.3.1: Output when the reset button was pressed after troubleshooting

3.3.2. Simulation Result Part 2



<u>3.3.2: Output when the timer and delay loop was running</u>

4. Conclusion

In conclusion, all the objectives for this lab were met. This lab was aimed towards familiarizing us with the Kinetis K60 Microcontroller as well as ARM assembly language. This lab comprised of two parts. The first of which asked us to determine the logic error in a given block of code. This logic error was fixed with an additional line of code and the source string was successfully copied to the destination variable. The main challenge faced in this part of the lab was when using the "myfprint" function as it manipulated the data in the register after printing it to the terminal. Through frequent debugging the workings of the function were figured out and hence we were successfully able to fix the code error.

The second and more complex part as it included a timer. This section of the lab was very thorough and exhaustive as we were required to print to the screen after every ten seconds, so the exactness of our timer was of utmost importance. We were successful in our implementation of this as time in-between was simulated by using various delay loops and through trial and error we were able to get it to about 9.95 seconds exactly. The proof of this fruitful execution can be seen under Simulation Results.

References

[1] Radu Muresan, "ENGG3640: Microcomputer Interfacing Laboratory Manual, Version 2", University of Guelph, July 2016.

Appendices

A. Main.c

main.c*

```
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUT
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPO
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTO
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQU
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
 * ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF TH
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
 ///////////////////////////////////////////////////////////////////
//  Includes
////////////////////////////////////////////////////////////////////

// Standard C Included Files
#include <stdio.h>
// SDK Included Files
#include "board.h"
#include "gpio_pins.h"
#include "fsl_debug_console.h"

extern void asmmain (void)
void myfprint(char *d)
{
  PRINTF ("%s\r\n", d);
}
int main(void)
{

  hardware_init();
  assmain();
}



/*****************************************************************
 * EOF
 *****************************************************************

```

Figure A.1: Page 1 main.c

B. myMain.s – Part 1

```
    PRESERVE8
    AREA MyCode, CODE, READONLY
    EXPORT asmmain
asmmain
    import myfprint
        LDR r0, =srcstr
        BL myfprint
        LDR r0, =dststr
        BL myfprint
        LDR r0,=dststr
        LDR r1, = srcstr

strcopy
    LDRB r3, [r1],#1
    STRB r3, [r0],#1
    CMP r3, #0
    BNE strcopy
    LDR r0, =srcstr
    BL myfprint
    LDR r0, =dststr
    BL myfprint
stop B stop

    ALIGN
    AREA MyData, DATA, READWRITE
srcstr DCB "String source", 0
dststr DCB "String destination",0
    END
```

Figure B.1: Page 1 myMain.s

Scopy.s* main.c

```
PRESERVE8
AREA   SCopy,   CODE,   READONLY
EXPORT strcopy
strcopy
LDRB  r2,   [r1],   #1
STRB  r2,   [r0],   #1
CMP  r2,   #0
BNE  strcopy
BX  lr
END
```

Figure B.2: Page 1 Scopy.s

C. myMain.s – Part 2

mymain.s | main.c*

```
    PRESERVE8
    AREA MyCode, CODE, READONLY
    EXPORT asmmain

asmmain
 import myfprint ;*d pointer is in register r0
 LDR r5,=0 ;counter register

tim
    LDR r1,=0xF2EDE0 ;Roughly one second @ 50Mhz
  ;previous line sets register 1 to about 1 second. Value in hex represents the time.
  ;must change value to get accurate time for the code you made yourself


Dloop SUB r1, #1 ;loop function that compares timer value to 0, wasting some time and
  NOP
  CMP r1,#0
  BNE Dloop
                ;if it not equal subract 1 from register 5
 ADD r5, #1

;These are all your compares for sets of ten seconds
 CMP r5, #10
 BEQ tens
 CMP r5, #20
 BEQ twentys
 CMP r5, #30
 BEQ thirtys
 CMP r5, #40
 BEQ fortys
 CMP r5, #50
 BEQ fiftys
 CMP r5, #60
 BEQ sixtys

 B tim

;these are your labels to what wyou want for each set of ten seconds
tens
 LDR r0, =tenstring
```

Figure C.1: Page 1 myMain.s

```
 B tim

;these are your labels to what wyou want for each set of ten seconds
tens
 LDR r0, =tenstring
 BL myfprint
 B tim
twentys
 LDR r0, =twentystring
 BL myfprint
 B tim
thirtys
 LDR  r0, =thirtystring
 BL myfprint
 B tim
fortys
 LDR  r0, =fortystring
 BL myfprint
 B tim
fiftys
 LDR r0, =fiftystring
 BL myfprint
 B tim
sixtys
 LDR  r0, =sixtystring
 BL myfprint
 LDR r5, =0
 B tim

 ALIGN
 AREA MyData, DATA, READWRITE
tenstring DCB "10s", 0
twentystring DCB "20s", 0
thirtystring DCB "30s", 0
fortystring DCB "40s", 0
fiftystring DCB "50s",0
sixtystring DCB "60s",0

END
```

Figure C.1: Page 2 myMain.c