



## Title: INTRODUCTION TO KEIL $\mu$ VISION-4 AND PROGRAMS ON DATA TRANSFER INSTRUCTIONS

Add two immediate values in the registers and store the result in third register.

### Program:

AREA RESET, DATA, READONLY

EXPORT \_\_Vectors

\_\_Vectors

DCD 0X10001000

DCD Reset\_Handler

ALIGN

AREA mycode, CODE, READONLY

ENTRY

EXPORT Reset\_Handler

Reset\_Handler

MOV R0, #10

MOV R1, #3

ADD R2, R0, R1

END



## ARM assembly language module

- Extensible Linking Format (ELF) sections (defined by the AREA directive).
- Application entry (defined by the ENTRY directive).
- Program end (defined by the END directive).



## **AREA:**

The AREA directive tells the assembler to define a new section of memory.

The memory can be code (instructions) or data and can have attributes such as READONLY, READWRITE and so on.

This is used to define one or more blocks of indivisible memory for code or data to be used by the linker.

The following is the format:

AREA sectionname attribute, attribute, ...

The following line defines a new area named mycode which has CODE and READONLY attributes: AREA mycode, CODE, READONLY Commonly used attributes are CODE, DATA, READONLY, READWRITE, ALIGN and END

**READONLY:** It is an attribute given to an area of memory which can only be read from. It is by default for CODE. This area is used to write the instructions.

**READWRITE:** It is attribute given to an area of memory which can be read from and written to. It is by default for DATA.

**CODE:** It is an attribute given to an area of memory used for executable machine instructions. It is by default READONLY memory.

**DATA:** It is an attribute given to an area of memory used for data and no instructions can be placed in this area. It is by default READWRITE memory.



**ALIGN:** It is an attribute given to an area of memory to indicate how memory should be allocated according to the addresses.

When the ALIGN is used for CODE and READONLY, it is aligned in 4-bytes address boundary by default since the ARM instructions are 32 bit word.

If it is written as ALIGN = 3, it indicates that the information should be placed in memory with addresses of 23 , that is for example 0x50000, 0x50008, 0x50010, 0x50018 and so on.



**EXPORT:** The EXPORT directive declares a symbol that can be used by the linker to resolve symbol references in separate object and library files.

**DCD (Define constant word):** Allocates a word size memory and initializes the values. Allocates one or more words of memory, aligned on 4-byte boundaries and defines initial run time contents of the memory.


**ENTRY:** The ENTRY directive declares an entry point to the program. It marks the first instruction to be executed. In applications using the C library, an entry point is also contained within the C library initialization code. Initialization code and exception handlers also contain entry points

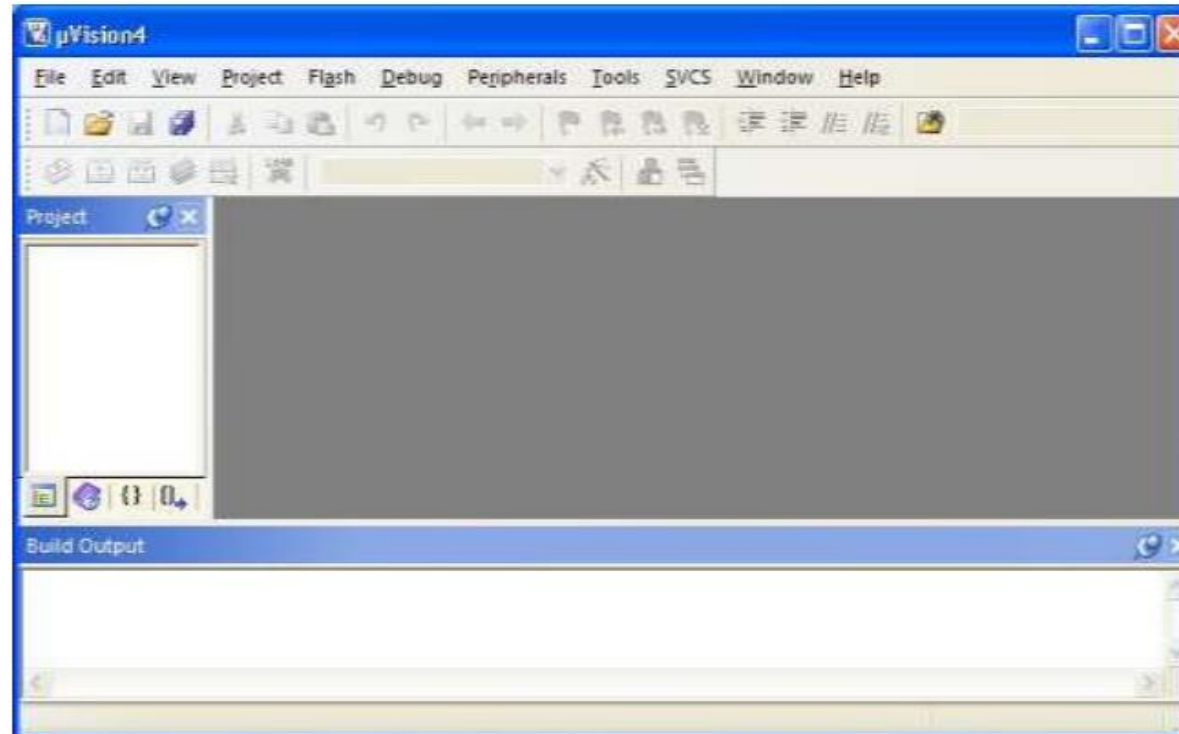
**END:** It indicates to the assembler the end of the source code. The END directive is the last line of the ARM assembly program and anything after the END directive in the source file is ignored by the assembler.



## I. Running an assembly language program in Keil u Vision 4

### Step 1:

- Create a directory with section followed by roll number (to be unique); e.g. A21
- Start up uVision4 by clicking on the icon  from the desktop or from the "Start" menu or "All Programs". The following screen appears.

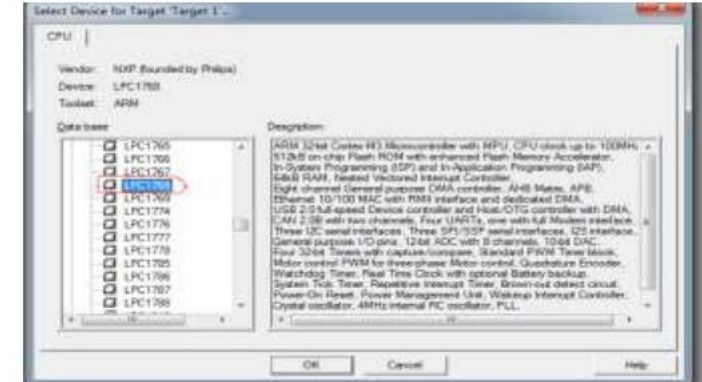




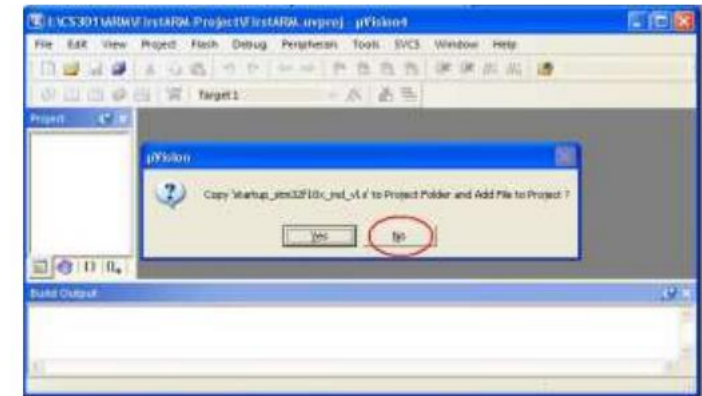


## Step 2: Create a project

To create a project, click on the "Project" menu from the uVision4 screen and select "New uVision Project". Then, select the folder you have created already, give project name and save.



Make sure you click on "NO" for the following pop up window.

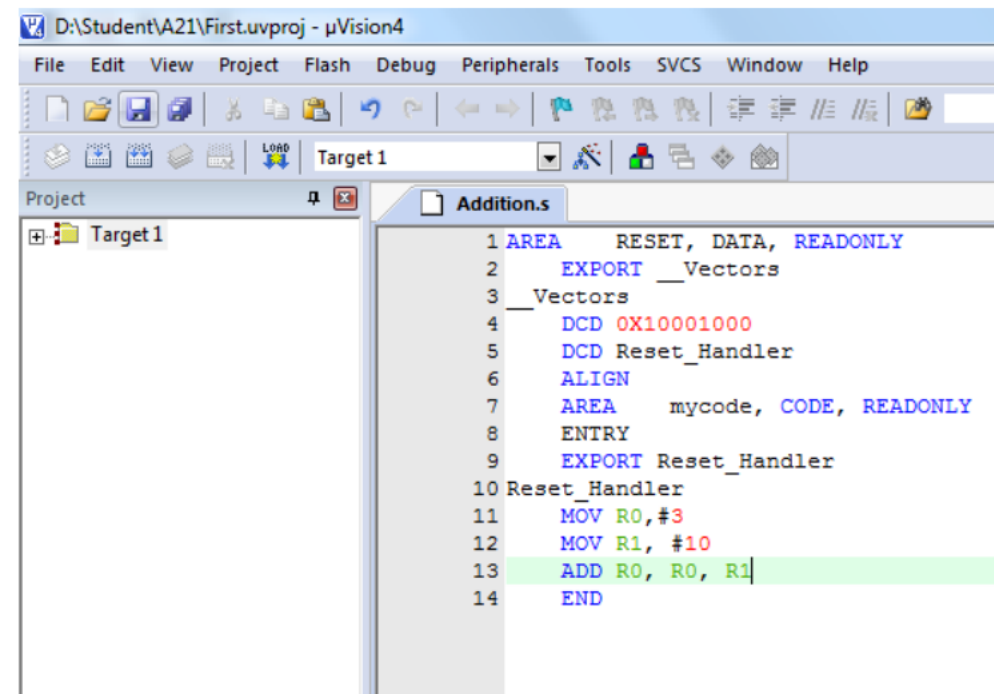
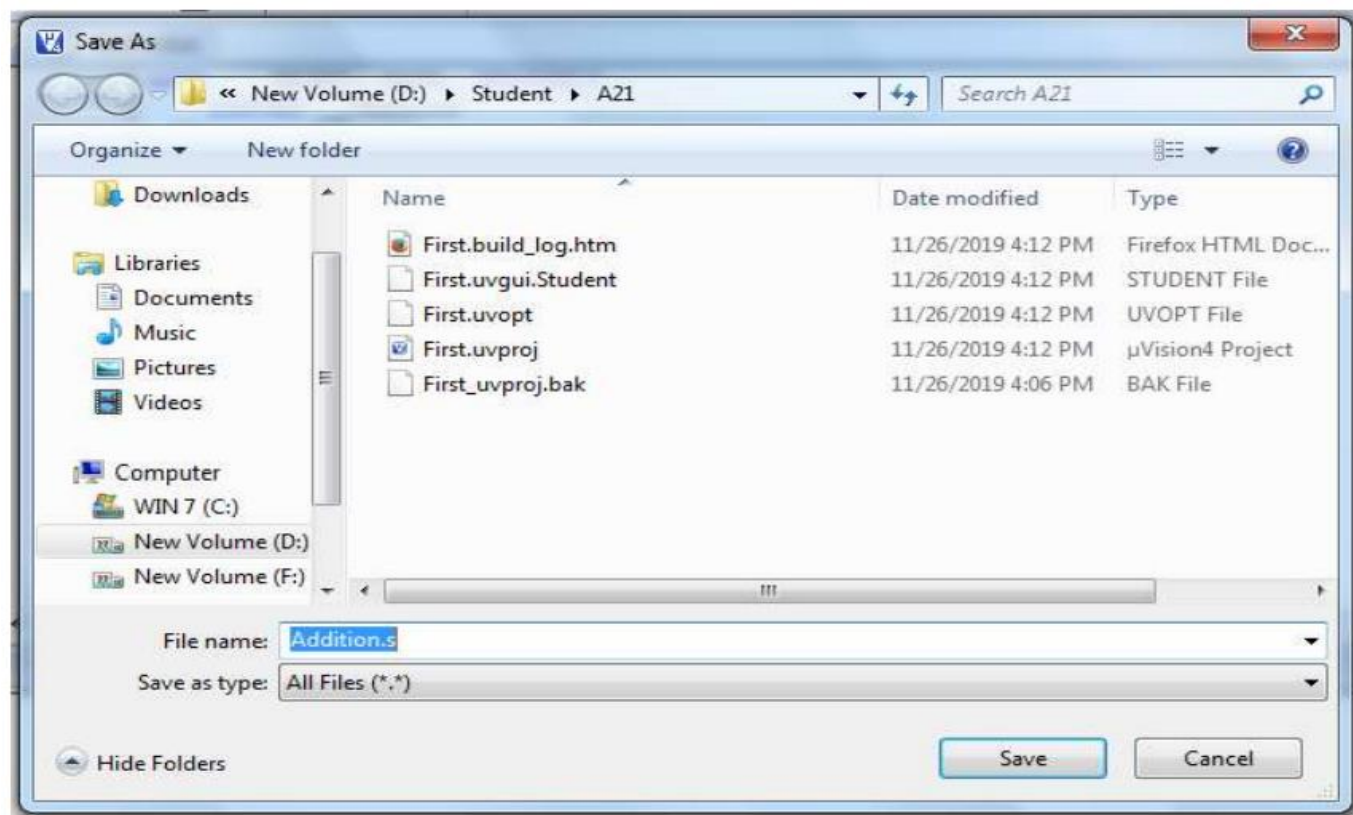


From the "Select Device for Target Target 1..." window, select "NXP" as the vendor. In that select LPC1768 ARM controller, then click on OK button. Some general information of the chip is shown in the description box.

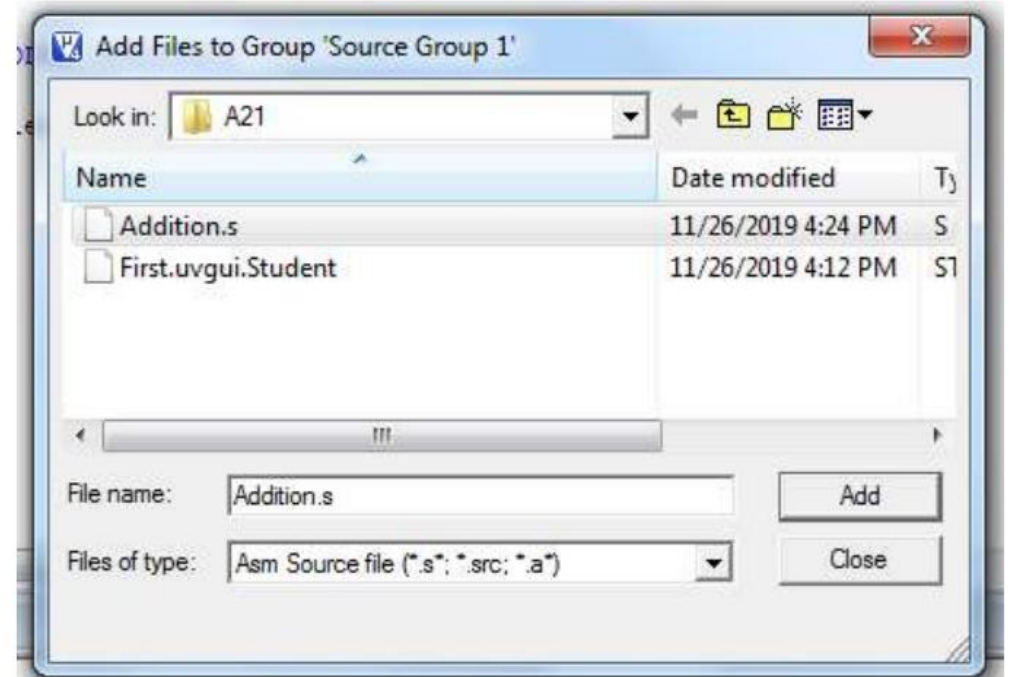
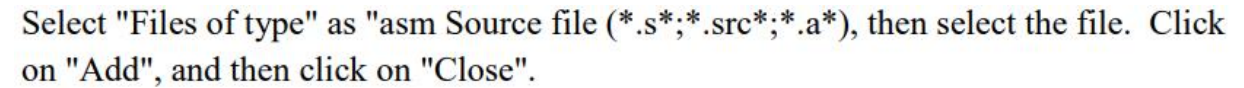


## Step 3: Create Source File

From the "File" menu, select "New", to get the editor window. Type the program here.  
(Note: give a tab space at the beginning). Save the program with .s extension in the directory.



Click on the + symbol near the Target 1 in the top left corner of the window. Right click on the "Source Group 1", select "Add Existing Files to Group 'Source Group 1'".





# MANIPAL INSTITUTE OF TECHNOLOGY

BENGALURU

(A constituent unit of MAHE, Manipal)

## Step 5: Build your project

Click on the "+" beside the "Source Group 1", you will see the program "Addition.s"  
Click on the "Build" button or from the "Project" menu, you will see the following screen.

The screenshot shows the uVision4 IDE interface. The title bar indicates the project path: D:\Student\A21\First.uvproj - uVision4. The menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations, debugging, and project management. The 'Project' window on the left shows a tree structure with 'Target 1' expanded, revealing 'Source Group 1' and the file 'Addition.s'. The main editor window displays the assembly code for 'Addition.s' with line numbers 1 through 14. The code includes area definitions, exports, and a reset handler function. The 'Build Output' window at the bottom shows the successful build of 'Target 1'.

```
1  AREA    RESET, DATA, READONLY
2  EXPORT  __Vectors
3  __Vectors
4  DCD 0X10001000
5  DCD Reset_Handler
6  ALIGN
7  AREA    mycode, CODE, READONLY
8  ENTRY
9  EXPORT Reset_Handler
10 Reset_Handler
11  MOV R0, #10
12  MOV R1, #3
13  ADD R0, R1
14  END
```

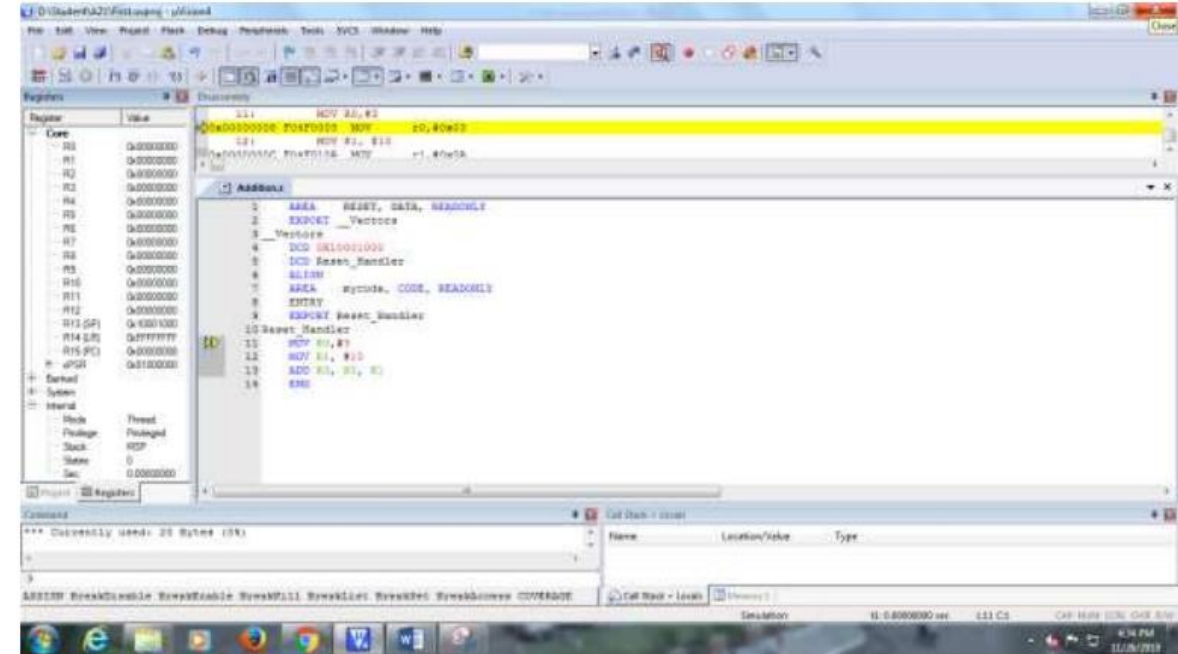
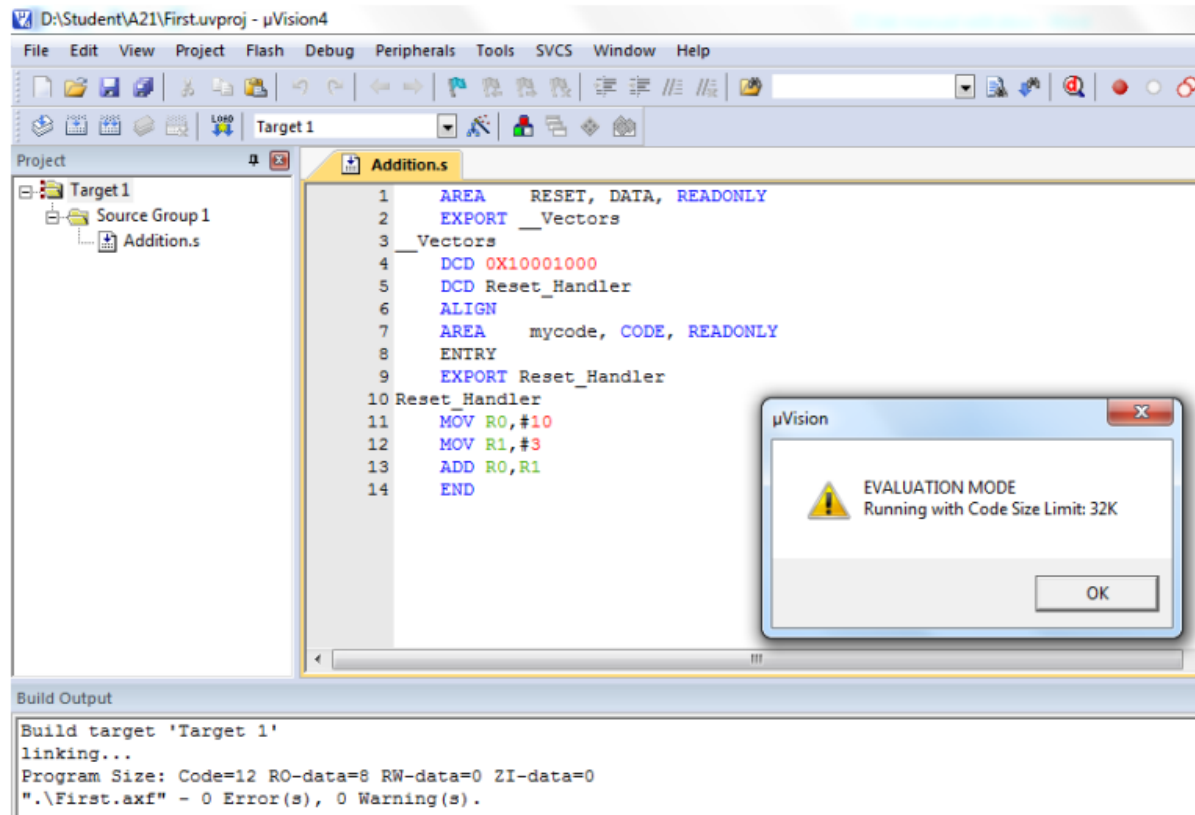
Build Output

```
Build target 'Target 1'
linking...
Program Size: Code=12 RO-data=8 RW-data=0 ZI-data=0
".\First.axf" - 0 Error(s), 0 Warning(s).
```



## Step 6: Run the program

Run the program through "Debug" menu.



Click on "OK" for the pop up window showing "EVALUATION MODE, Running with Code Size Limit: 32K". You will see the following window.



- Open uVision4 to full screen to have a better and complete view. The left hand side window shows the registers and the right side window shows the program code. There are some other windows open. Adjust the size of them to have better view. Run the program step by step; observe the change of the values in the registers.
- Run the program using the **Step Over** button or click on Step Over from the Debug menu. It executes the instructions of the program one after another.
- To trace the program one can use the **Step button**, as well. The difference between the **Step Over** and Step is in executing functions.
- While Step goes into the function and executes its instructions one by one, **Step Over** executes the function completely and goes to the instruction next to the function.
- To see the difference between them, trace the program once with **Step Over** and then with **Step**. When PC is executing the function and want the function to be executed completely one can use **Step Out**.
- In this case, the instructions of the function will be executed, it returns from the function, and goes to the instruction which is next to the function call.



