Engineering Software Research Center

www.esrcen.com esrc.nathanm7@gmail.com

Jl. Situ Aksan 29, Bandung 40221

Tel: (62)-21-6041685, Mobile: (62)-878-25670070

BALI V.1.0

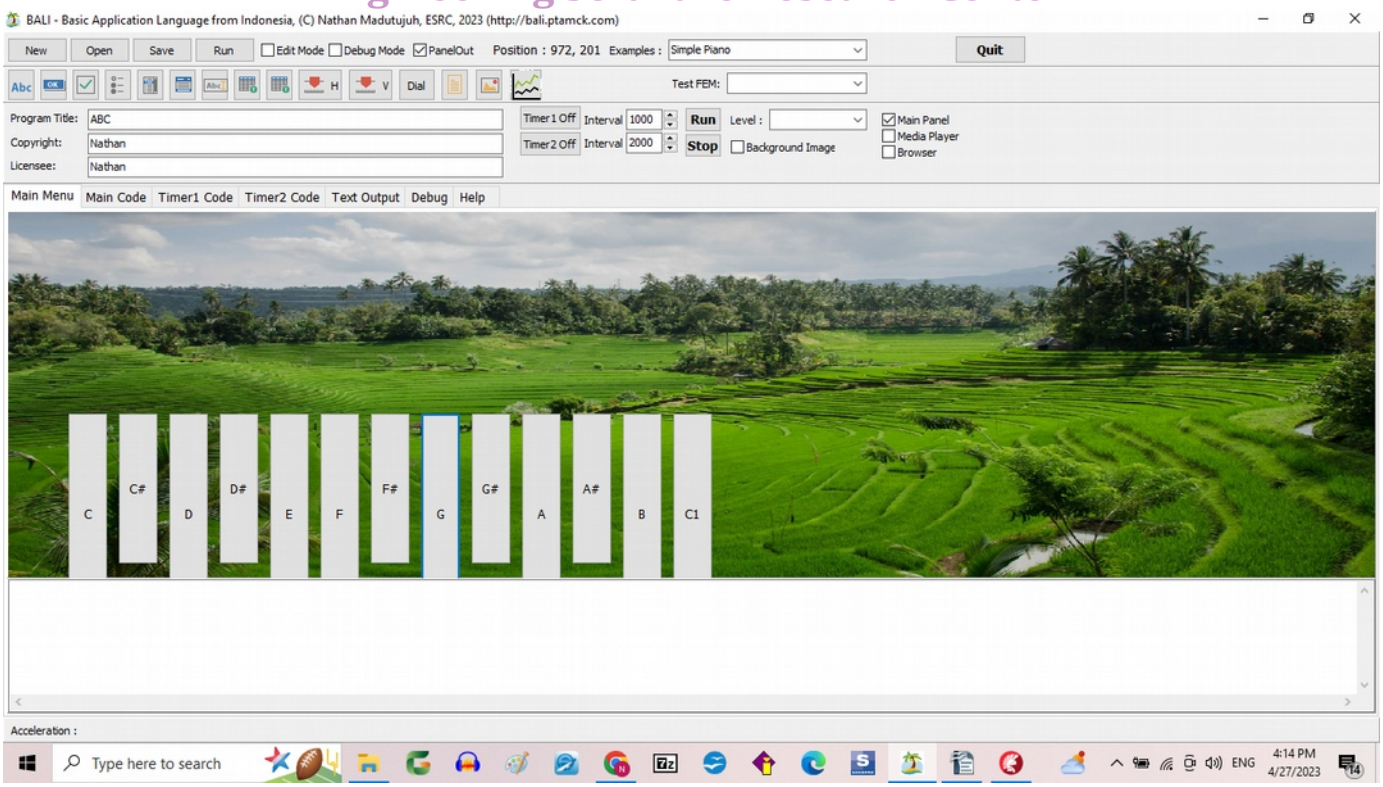
Basic Application Language from Indonesia

A general-purpose interactive programming language for coding class, science and engineering apps development

by

Dr. Nathan Madutujuh

Engineering Software Research Center



http://bali.ptamck.com

April 2023

PREFACE

BALI (Basic Application Language from Indonesia) is a new and unique programming language created by Dr. Nathan Madutujuh from ESRC, Bandung, Indonesia. BALI can be learned in very short time to create useful applications for various fields. The language is integrated with its development and run-time environment, so the development, testing and running of the application created can be done seamlessly.

BALI uses interpreter instead of compiler. But with the help of modern and fast CPU, the running time is quite fast. And also the language is very easy to extend by adding user functions and procedures. To make it easier to learn, the syntax of BALI language is a mix of C and Pascal syntax, taken the best of both languages.

BALI can be used for any level of students, from elementary to college students. It has several modules including Control structures, Timer, MIDI, Media Player, Text File, Graphics, String functions, Numerical functions, Vector and Matrix functions, Web Browser, Vector and Matrix operations, and a simple Finite element module by ESRC.

Also a unique feature in BALI is a user can use an object name and object parameters directly inside any expression (object name can be used for variable name), and if any value is assigned to an object name, it will be displayed automatically in object's visual part. This will enable a very intuitive and make it very easy to learn visual programming concept.

To make the learning curve faster, several examples and tutorials have been provided inside BALI development environment, so a thick and difficult to read manual is not needed anymore. Also an online community and offline communities in several cities will be provided as a platform to share and market the application created.

BALI can be afforded with a very low price, according to the modules integrated in the system. Hopefully using BALI, all level of students can learn and make a programming language easily and in short time. **Programming must be fun !**

Bandung, August 17, 2022

Dr. Nathan Madutujuh

TABLE OF CONTENTS

Introduction to BALI

BALI (**B**asic **A**pplication **L**anguage from **I**ndonesia) is a new programming language created by Dr. Nathan Madutujuh from ESRC, Bandung, Indonesia. The name is taken from famous Bali Island, the tropical paradise in Indonesia.

The main reasons to create another programming language are the need for a simple and easy to learn a modern, event-driven, bject oriented visual programming language that can be used for coding class for students from elementary to advanced level.

BALI can be used to create useful applications for various fields in a very short time. For example, the hello program can be created and run in less than 5 minutes. The piano program using MIDI modules can be created in 10 minutes.

The language is integrated with its development and run-time environment, so the development, testing and running of the application created can be done seamlessly. Although BALI uses interpreter instead of compiler, but with the help of modern and fast CPU, the running time is quite fast.

Also the language is very easy to extend by adding user functions and procedures. To make it easier to learn, the syntax of BALI language is a mix of C and Pascal syntax, taken the best of both languages. All modules and functions, including user-defined functions are compiled as library, to get maximum speed for procedures and functions that will take long time to run.

BALI has several modules as in a standard programming language, identifiers, variables, constants, expressions and statements, including logical and loop control structures, Timer, MIDI, Media Player, Text File, Graphics, String functions, Numerical functions, Vector and Matrix functions, Web Browser, and also a simple Finite element module by ESRC.

Using BALI, any elementary student can learn how to develop a visual program easily, and any scientist or engineer can develop a program prototype in a very fast time, without a long time learning curve as in other programming languages.

In other words, BALI is a combination of the stable and clear syntax of Pascal and C, the flexibility of Python, the Visual objects of C#, and the matrix and numerical features of MathCad.

BALI Programmer's Community

Because BALI will be distributed a very affordable price, according to the modules integrated in the system, it can be targeted to a very wide market. The license price is as follows:

Category	Level	Local Price	International Price
Elementary students	K1-K6	Rp. 50,000,-	USD 10
High school students	K7-K12	Rp. 100,000,-	USD 20
College students		Rp. 150,000,-	USD 30
Others		Rp. 250,000,-	USD 50

To make the learning curve faster, several examples and tutorials have been provided inside BALI development environment, so a thick and difficult to read manual is not needed anymore. BALI user's can register themselves to BALI Programmer's Community website, to be able to communicate and share to each other, and also sale or distribute the developed applications to other BALI users. In the community, useful tutorials and sample applications will be provided and shared.

Also several BALI coding classes will be established in major cities in Indonesia, this will create job

and business opportunities for young people in Indonesia and other countries. Using BALI, all level of students can learn and make a programming language easily and in short time.
Programming must be fun !

A BALI Program

A BALI program is a collection of nonvisual and visual objects. Program's code can be placed at main panel code, any timer code, and any object event handler's code. There are 2 default timers available that can be used to control the execution of each part of code. A visual object can be added, renamed, and set its parameters accordingly. If a visual object is also reflecting a variable, the name of the visual object should be the variable name.

BALI Program Features

BALI has many features usually found in modern visual computer languages. Among them are visual objects, event driven programming, timers, MIDI music, control structures, strings, files, math, graphics, vector and matrix. Also a unique feature in BALI is a user can use an object name and object parameters directly inside any expression (object name can be used for variable name), and if any value is assigned to an object name, it will be displayed automatically in object's visual part. This will enable a very intuitive and an easy to learn visual programming environment.

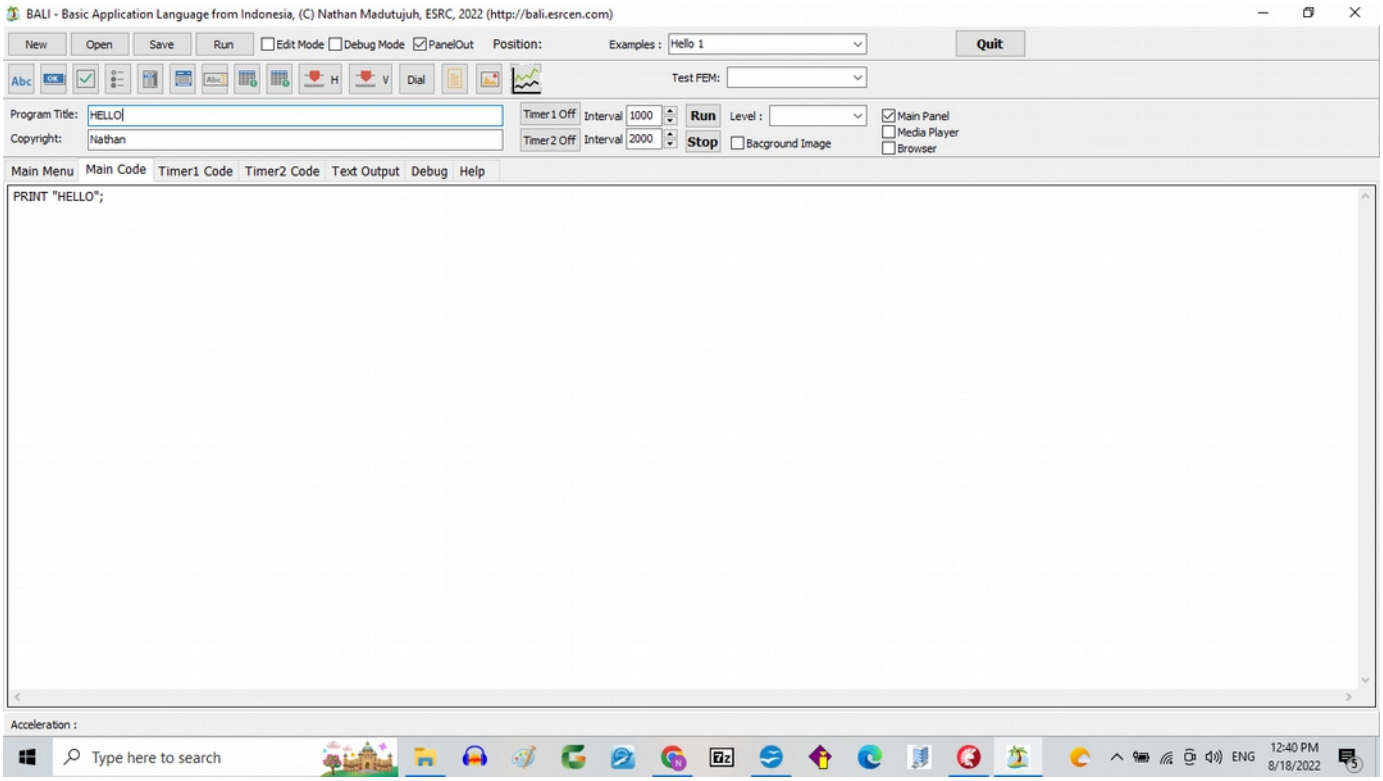
BALI language features are :

- Visual programming environment with integrated IDE and Run-time
- Fast and Easy to learn for all ages level ("Hello" program in 5 minutes)
- Low-cost hardware requirement
- For coding class or engineering applications
- Interpreter and Compiler language
- Interactive and Dynamic Object Oriented Programming :
  Label, Button, Checkbox, Edit, Select, Listbox, Combobox, Trackbar, Dial, Memo, Image, Calendar, MediaPlayer, Browser, Vector, Matrix
- Object name can be used as variable
- User editable object parameters
- Visual Programming with IDE
- Built-in Timers and separate timer event handler for each object
- Event-driven programming with user defined handlers
- Mixed C and Pascal syntax
- No Variable declaration
- No semi-colon required
- = for assignment, == for equality checking
- script for event handler, extendable functions
- User defined extendable library
- Strings and Keyboard Functions
- Integer and Floating Number Math Functions
- MIDI Module for Kids
- Graphics Module
- Text File module
- Database Module : SQLite*
- Vector and Matrix Module
- SANS Finite Element Module
- Location, Acceleration Sensors
- Data Acquisition Module*
- Multiple platform support

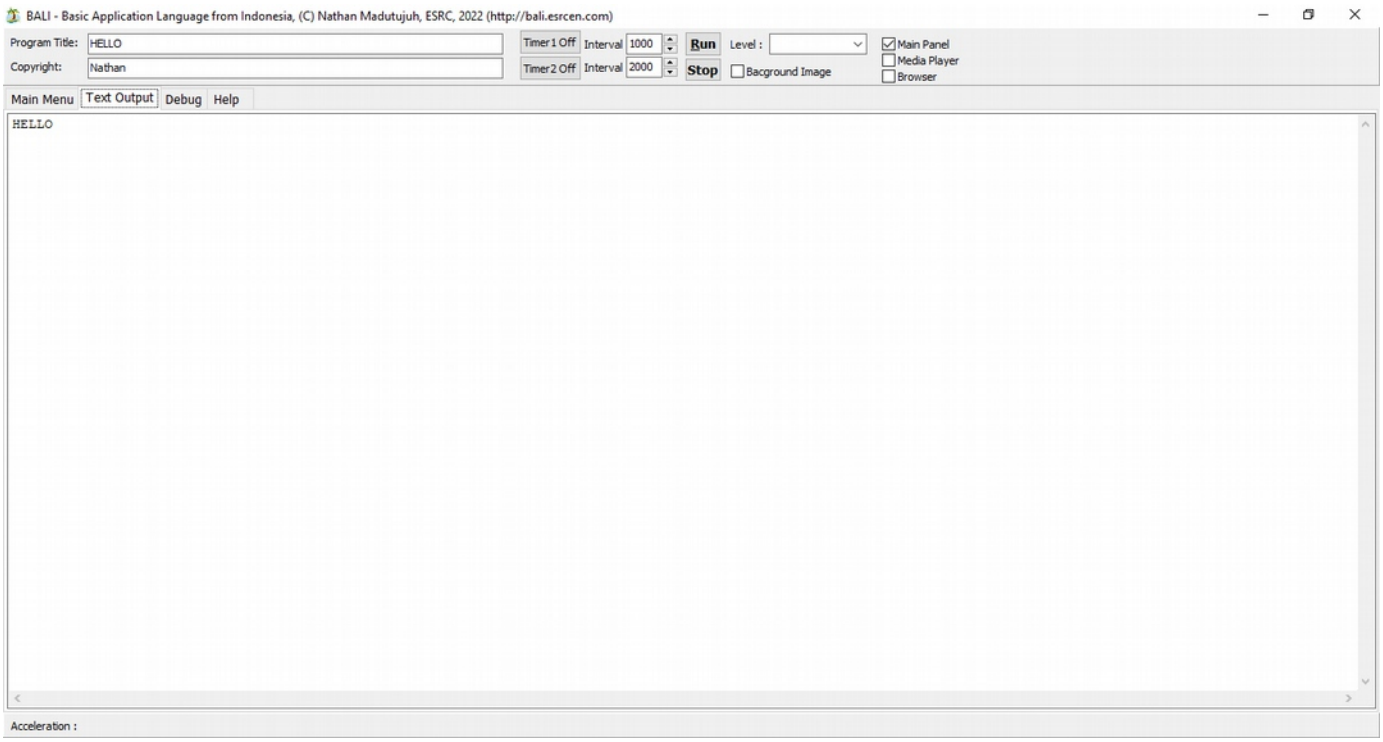
Non-visual Programming Tutorial

1. Hello Program

Enter : PRINT “HELLO !”; at the main code

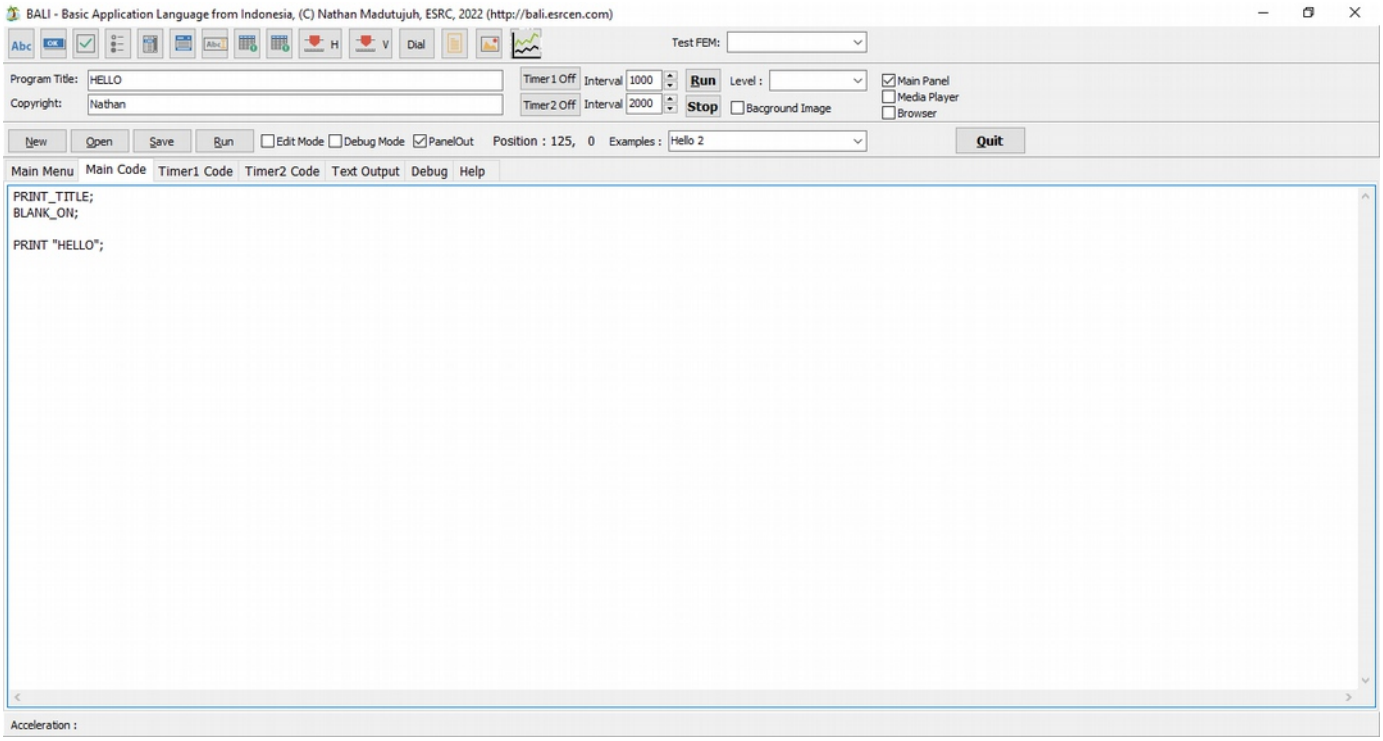


Click [RUN], then the string “HELLO !” will appear at Tab Text Output as follows:

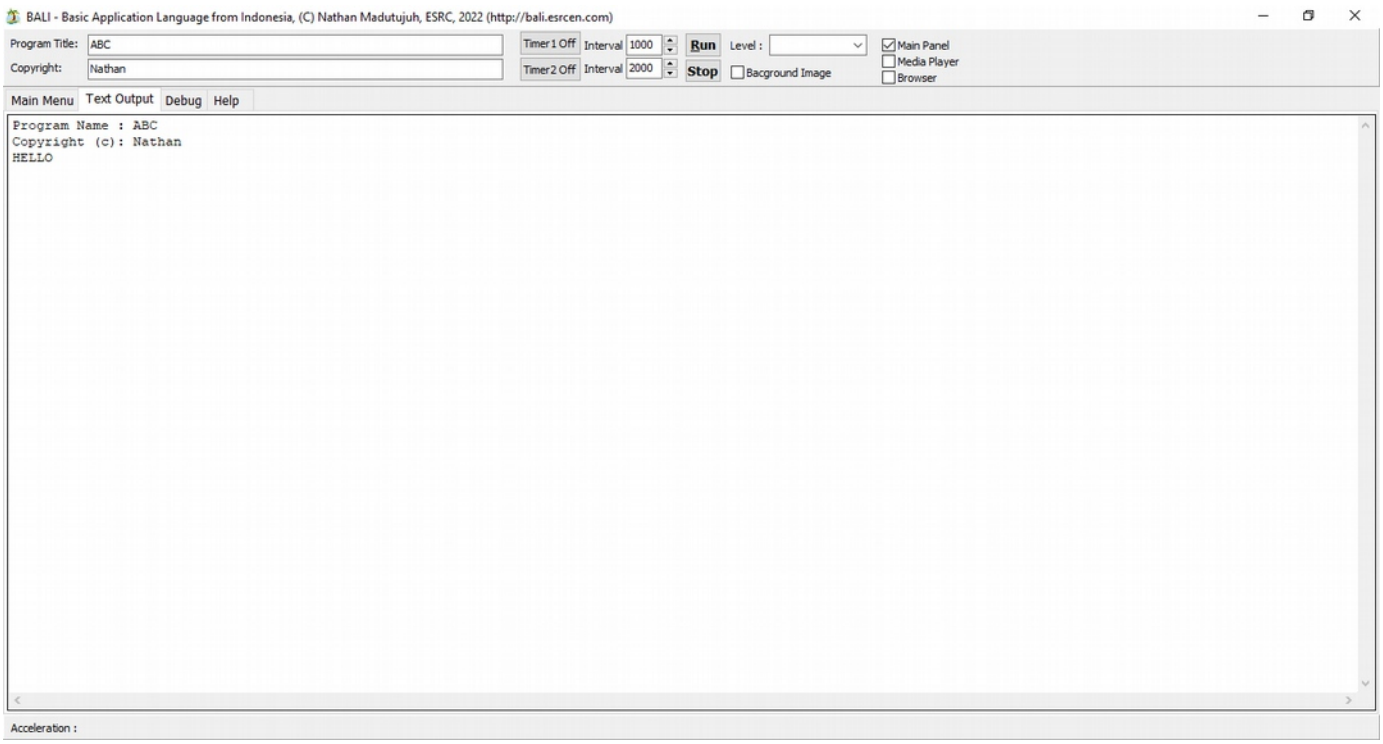


Another variation: add the following to the main code:

```
PRINT_TITLE;
BLANK_ON;
PRINT "HELLO";
```



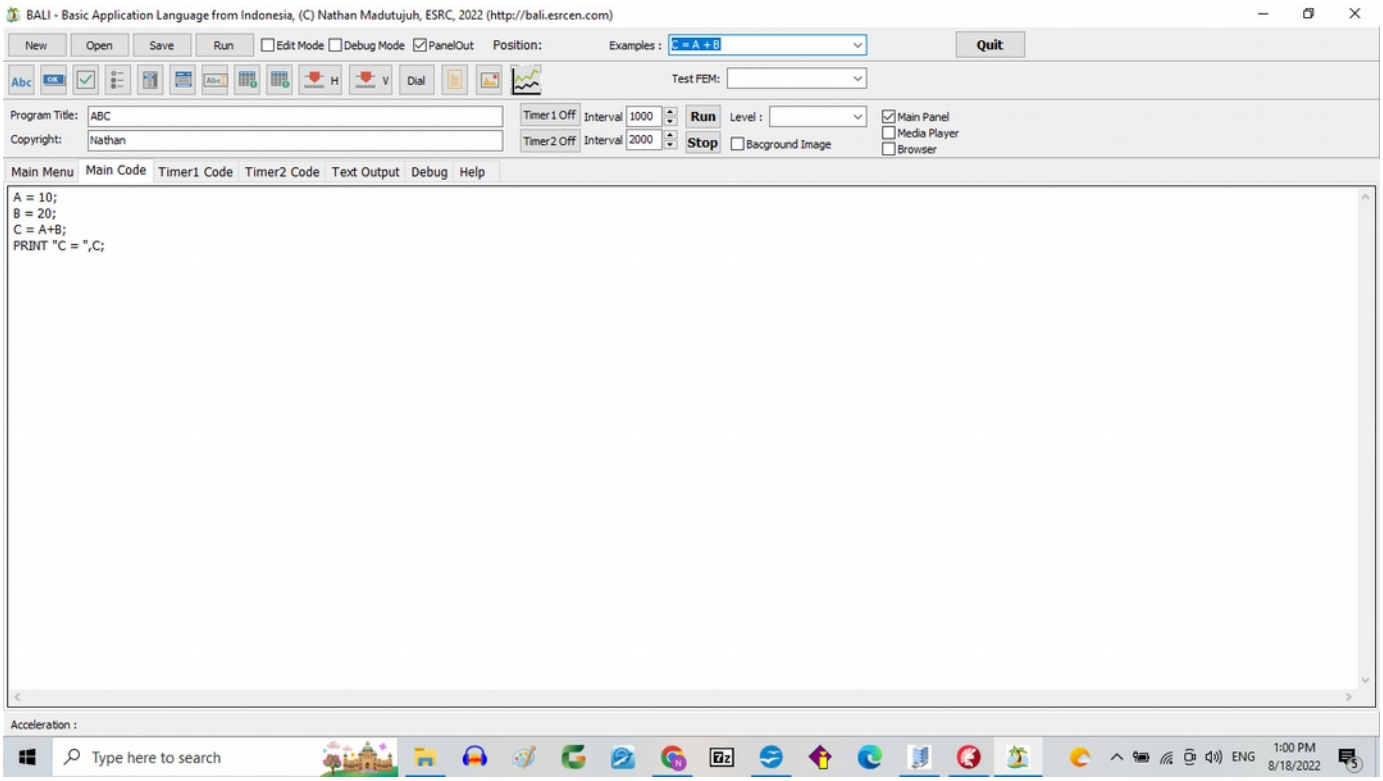
The the output will be :



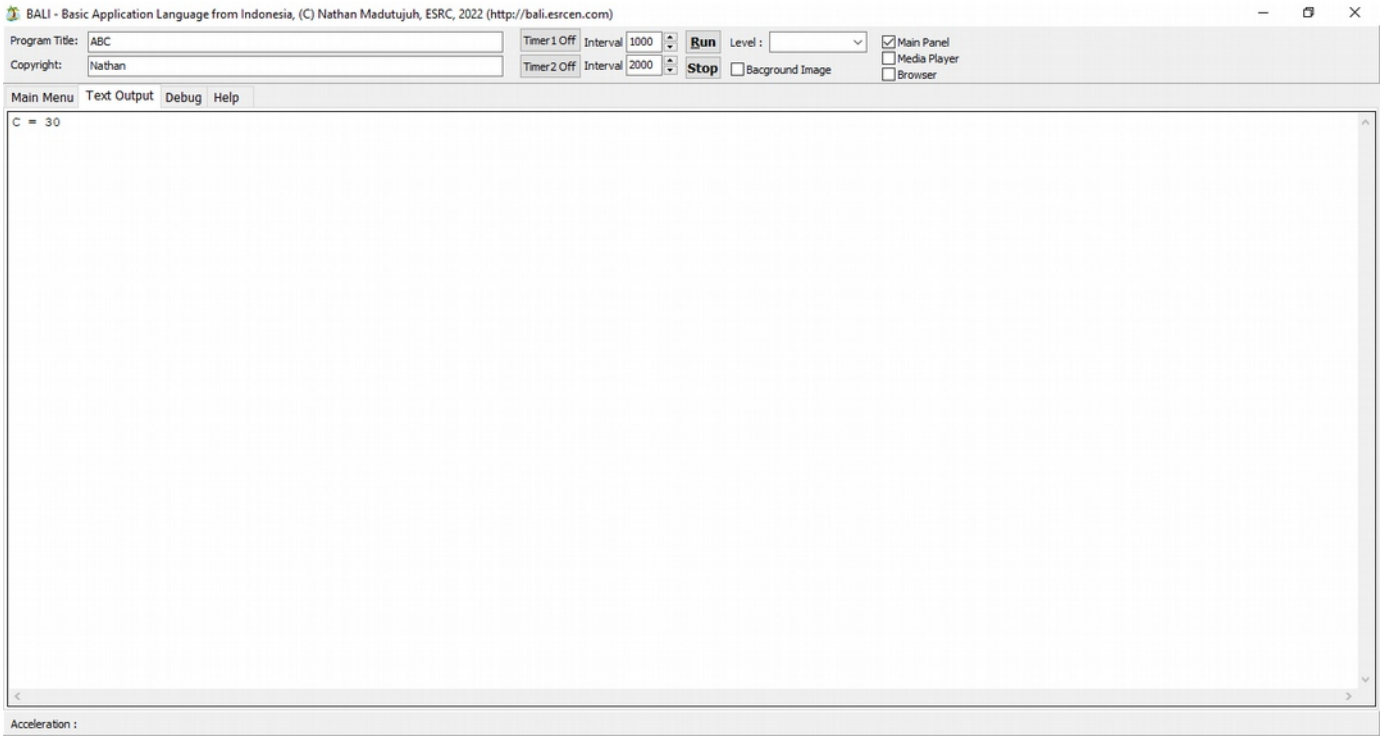
2. C = A + B

Here we will create a simple program that adds two numbers A and B, and display it as C.Another Add the following to the main code:

```
PRINT_TITLE;
BLANK_ON;
A = 10;
B = 20;
C = 30;
PRINT "C = ", C;
```

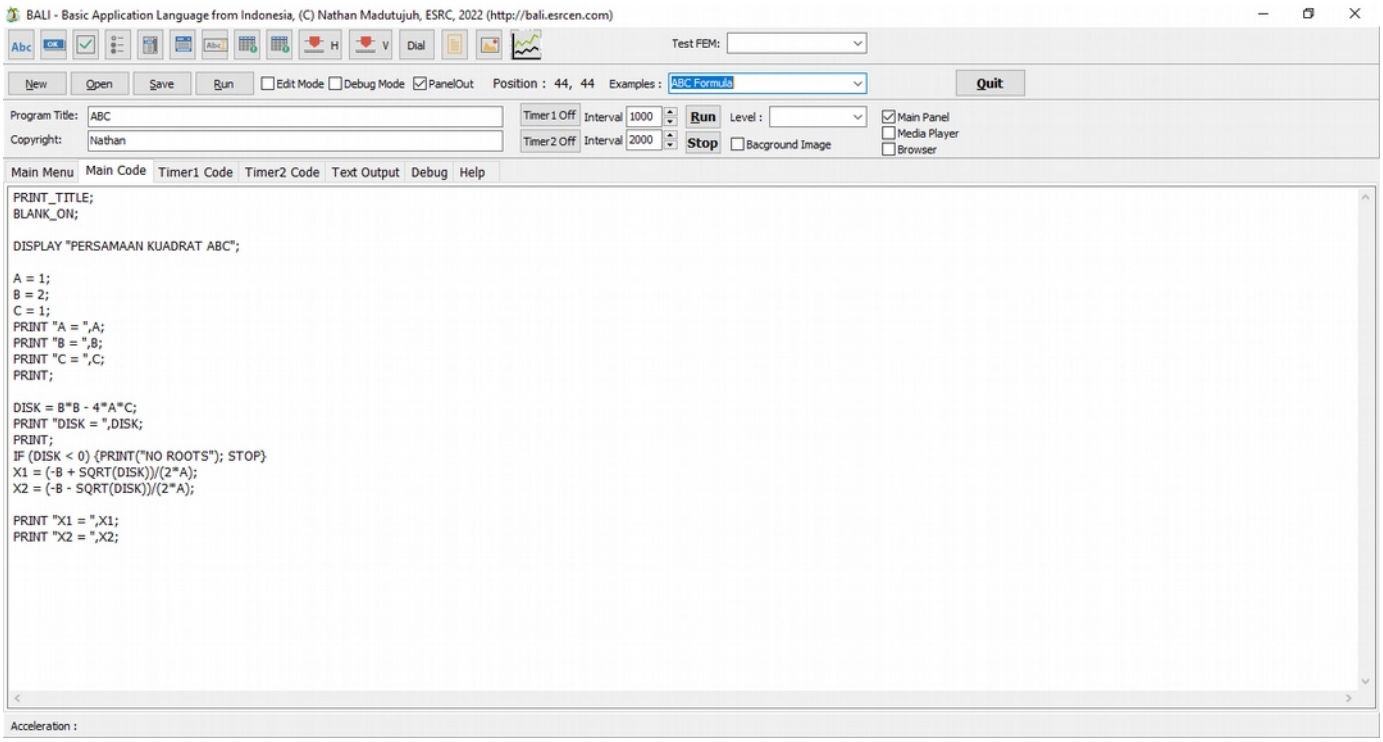


The text output will be :

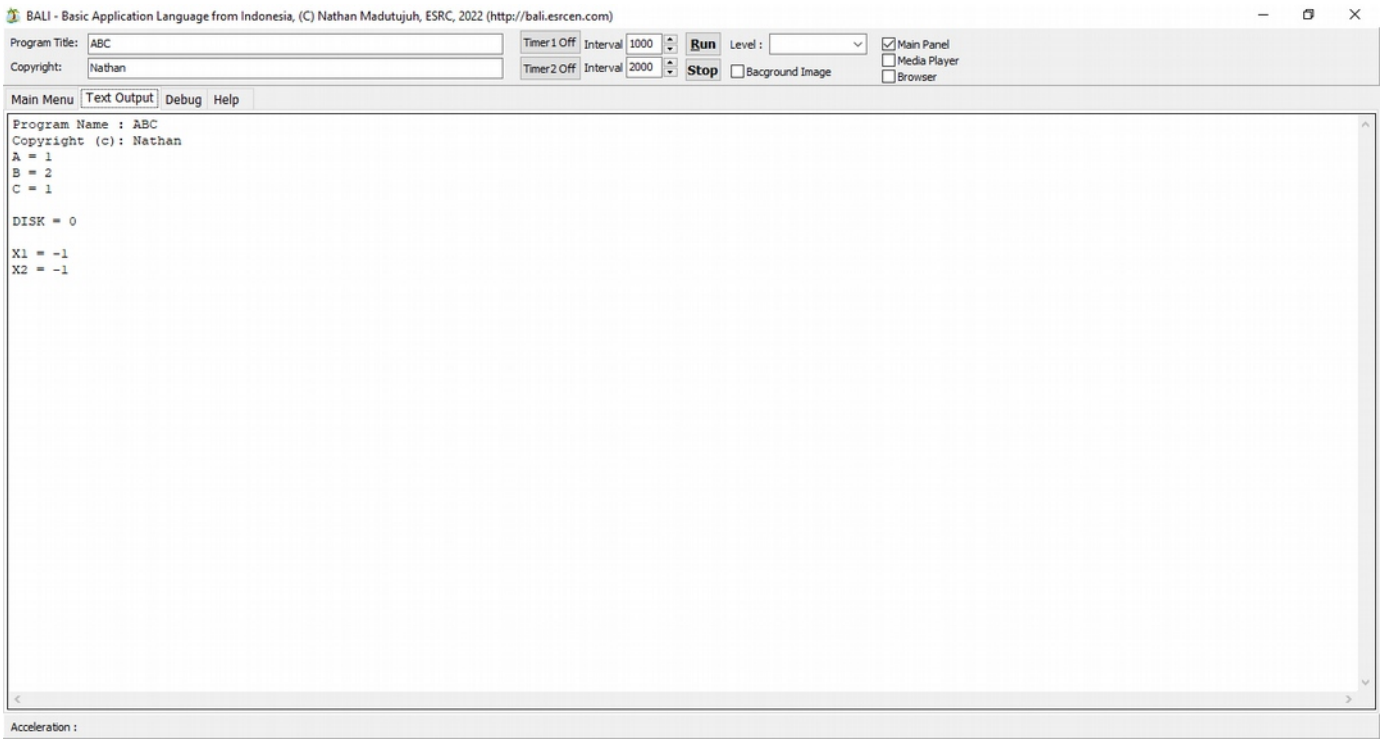


3. ABC Formula

Find roots of a quadratic equation : A*X^2 + B*X + C = 0



The text output will be:



Visual Programming Tutorial:

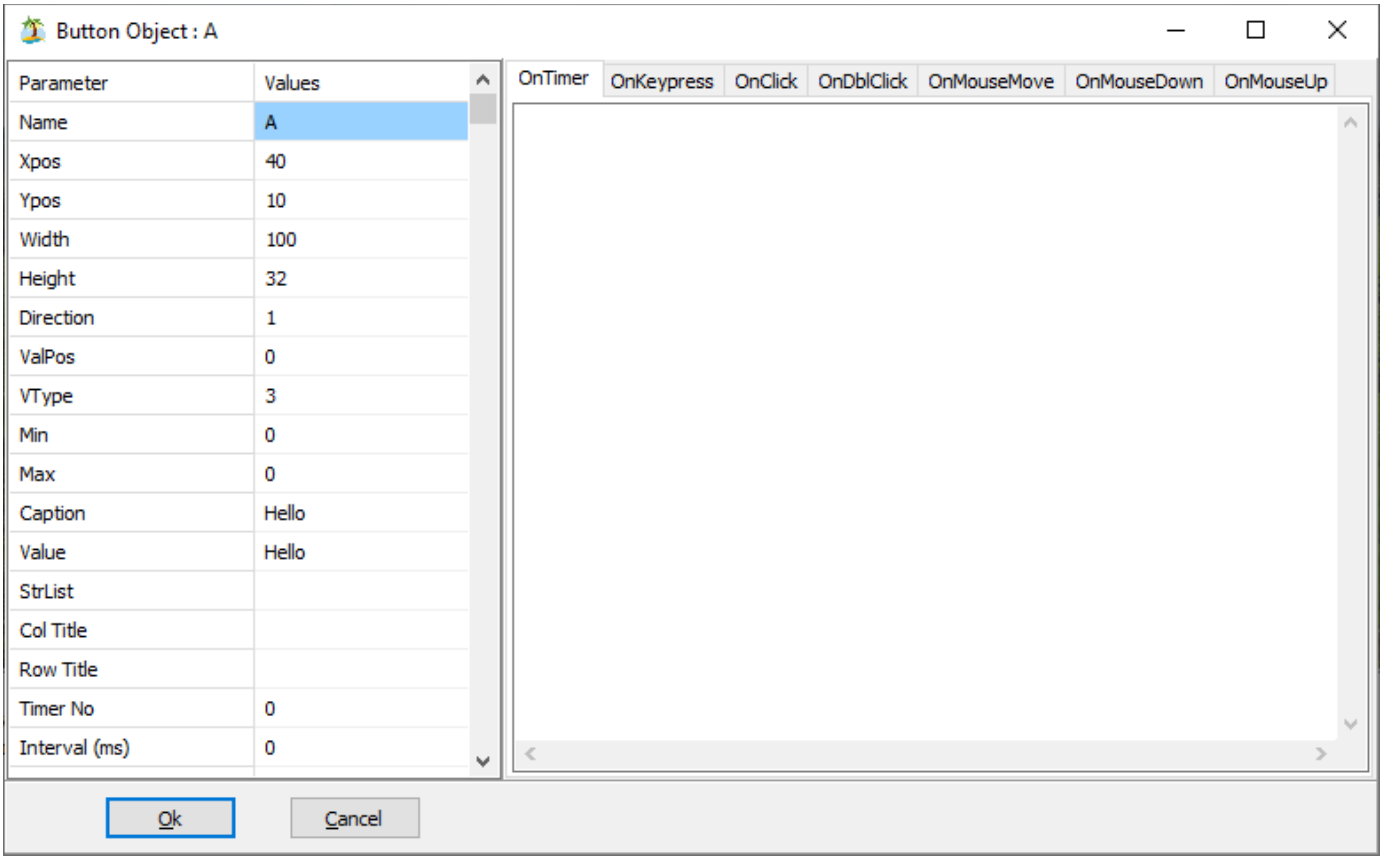
1. Visual Programming concepts

A visual program consists of several visual objects with its own parameters and event handler, main code and separate code for each timer if used.
Event handler is a part of code to be executed when an event happened.
An example of an event are : Keyboard clicked, object clicked, object double clicked, mouse down, mouse up, mouse move, etc.

Certain object can have value that can be used directly in any expression using its name as variable.
If only the name of the object is used as variable, the value will be default according to the type of the object. Other object parameters can be accessed also by using the following syntax:

OBJECTNAME.parametername
A.Left
A.FontSize

Object parameters can be changed during edit mode interactively, through the Object Parameters Window:



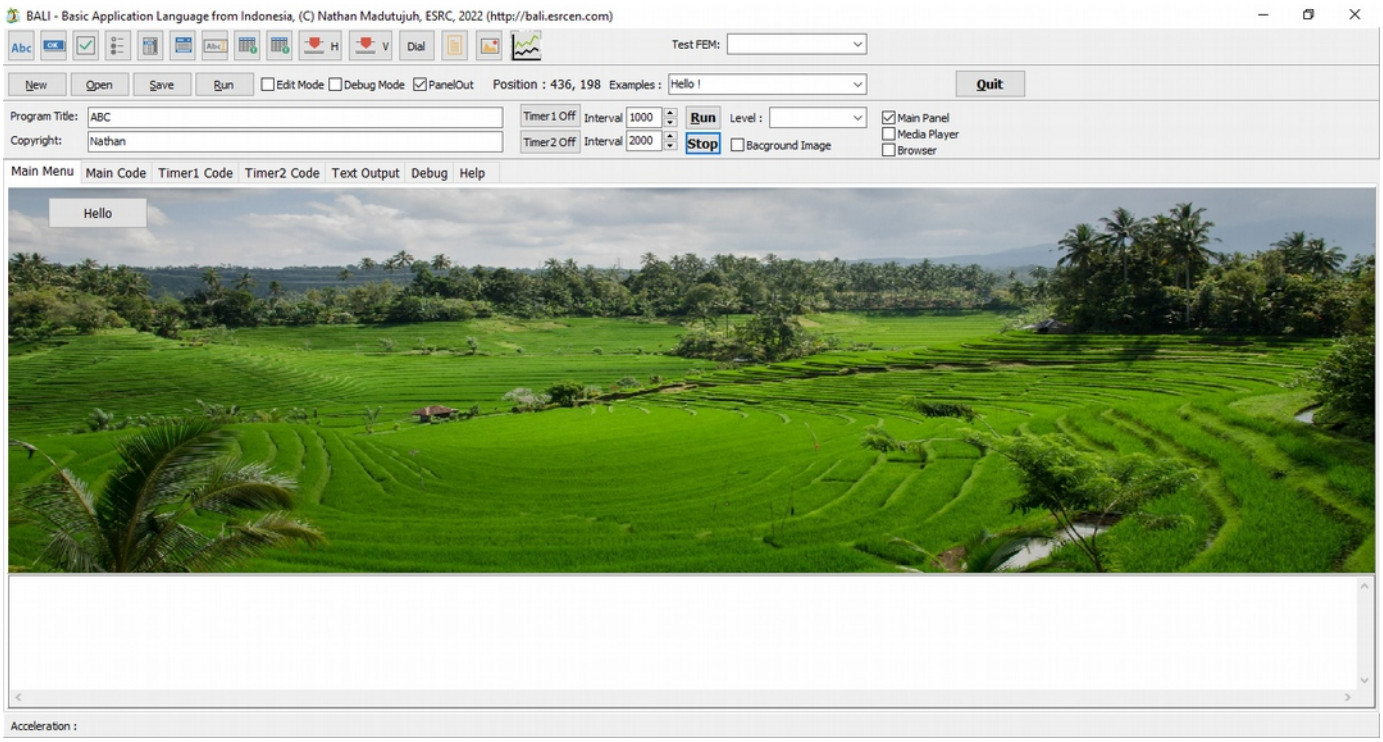
Object parameters can be changed also during running time by using the above format. For example:

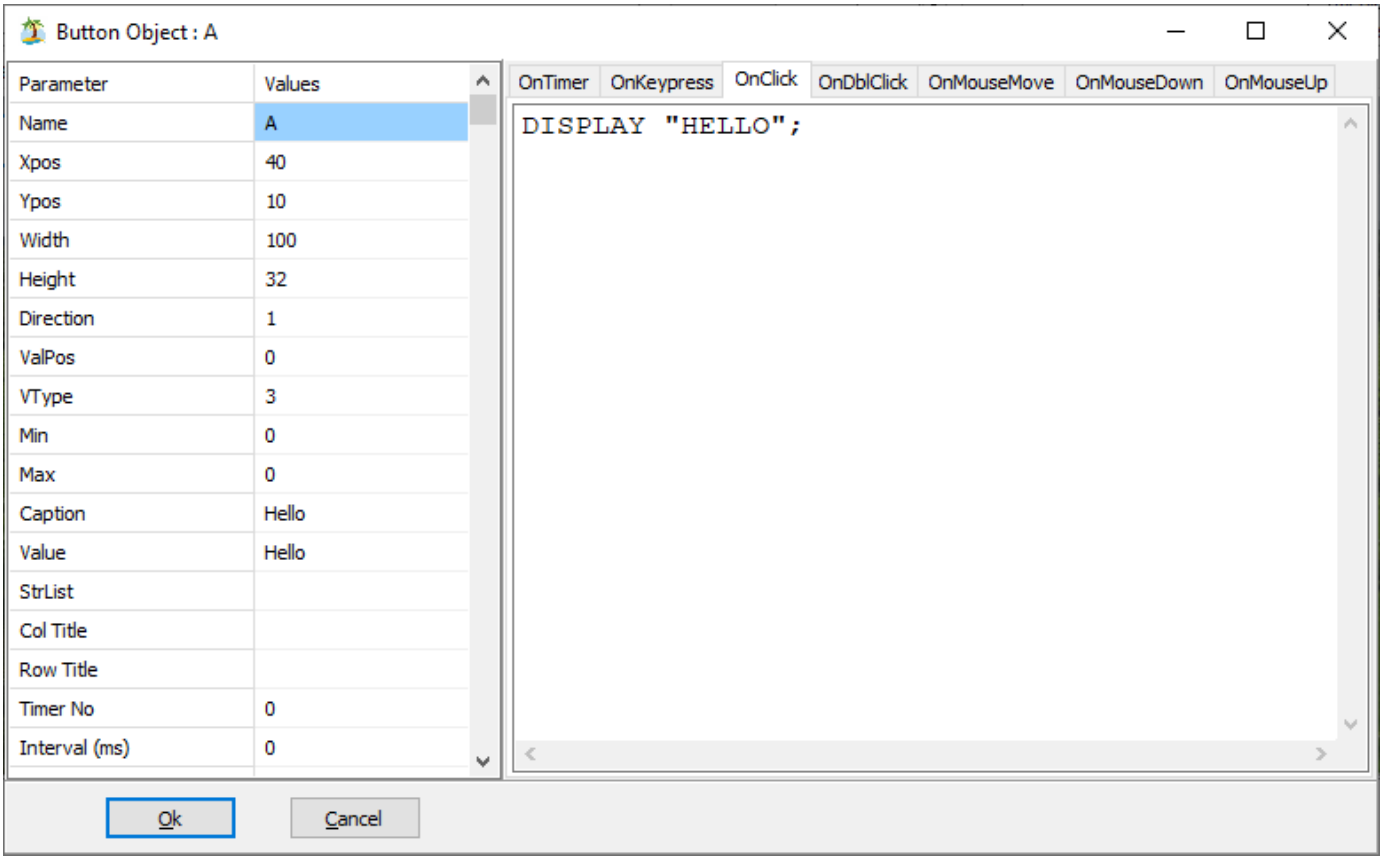
A.Left = X + 10;

2. Hello Program

Add a button named 'Hello'.
Add a text at event handler OnClick:

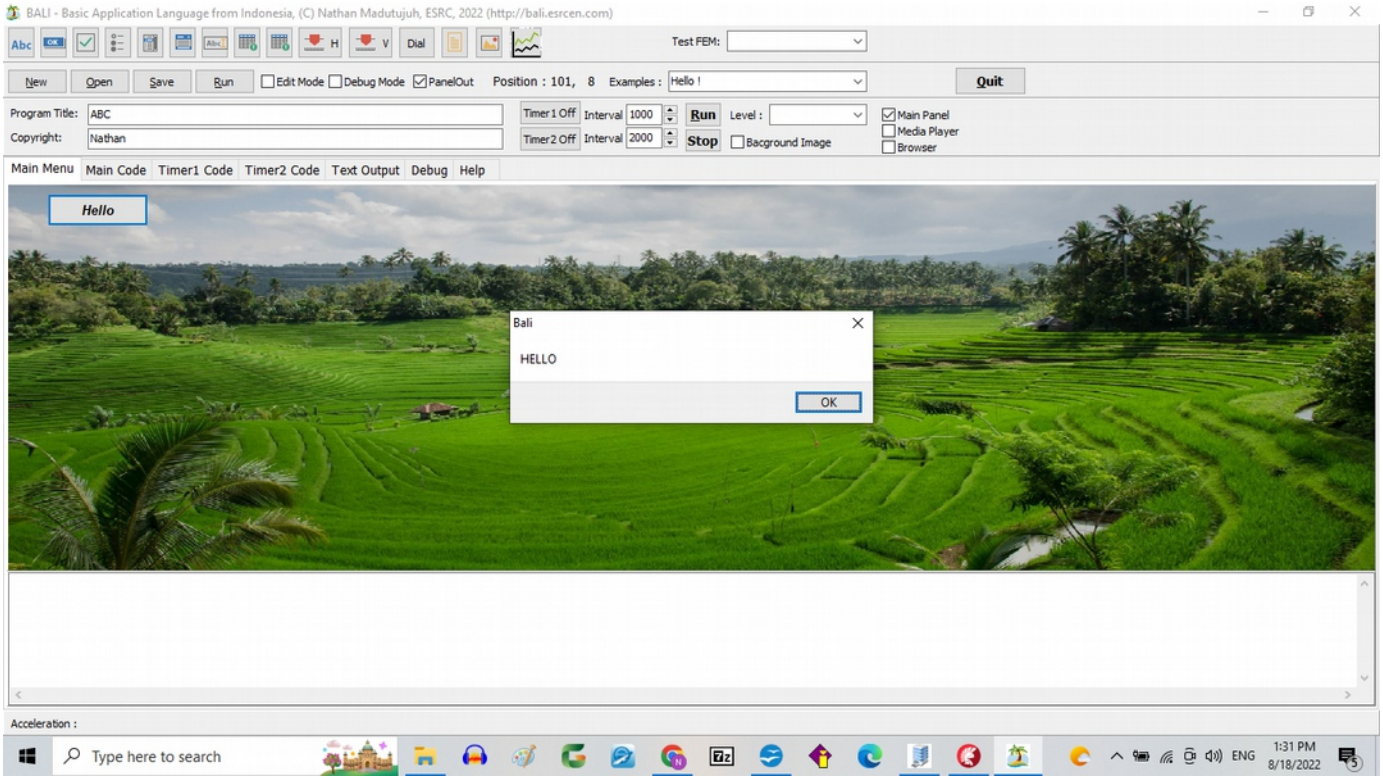
```
DISPLAY “HELLO !”;
```





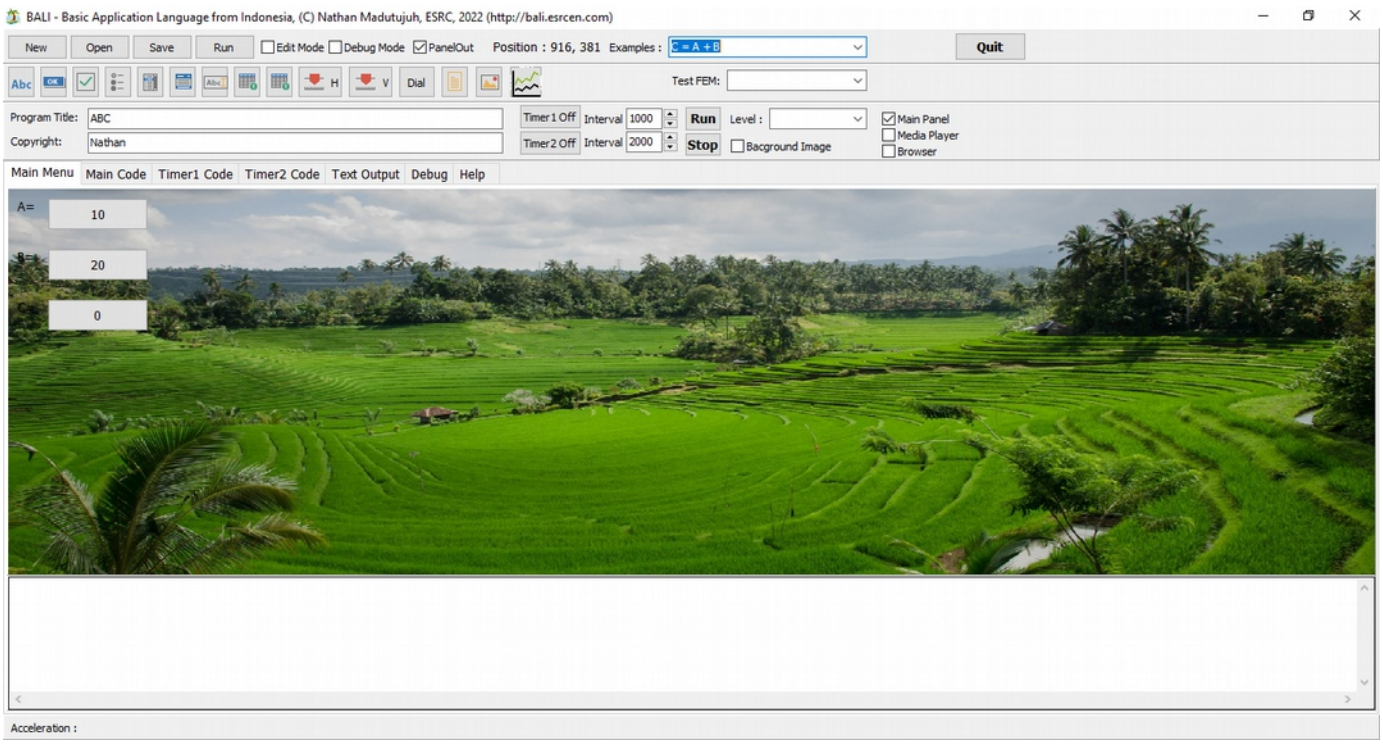
When the button is clicked, the message “HELLO !” will be displayed :


Bali
HELLO
OK

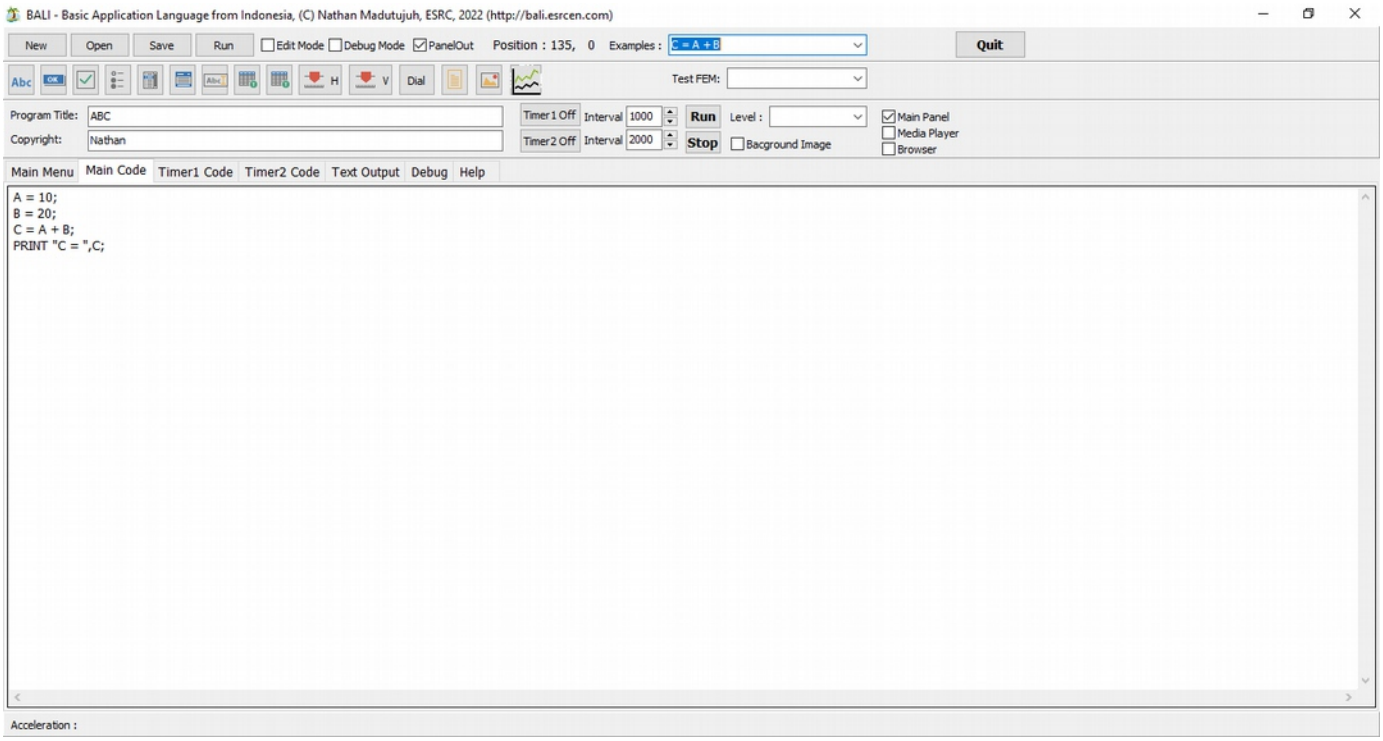

BALI - Basic Application Language from Indonesia, (C) Nathan Madutujuh, ESRC, 2022 (http://bali.esrcen.com)
Test FEM:
New
Open
Save
Run
Edit Mode
Debug Mode
PanelOut
Position : 101, 8
Examples :
Hello !
Quit
Program Title:
ABC
Copyright:
Nathan
Timer1 Off
Interval
1000
Timer2 Off
Interval
2000
Run
Stop
Level :
Bacground Image
Main Panel
Media Player
Browser
Main Menu
Main Code
Timer1 Code
Timer2 Code
Text Output
Debug
Help
Hello
Bali
HELLO
OK
Acceleration :
Type here to search
ENG
1:31 PM
8/18/2022

3. Program: C = A + B

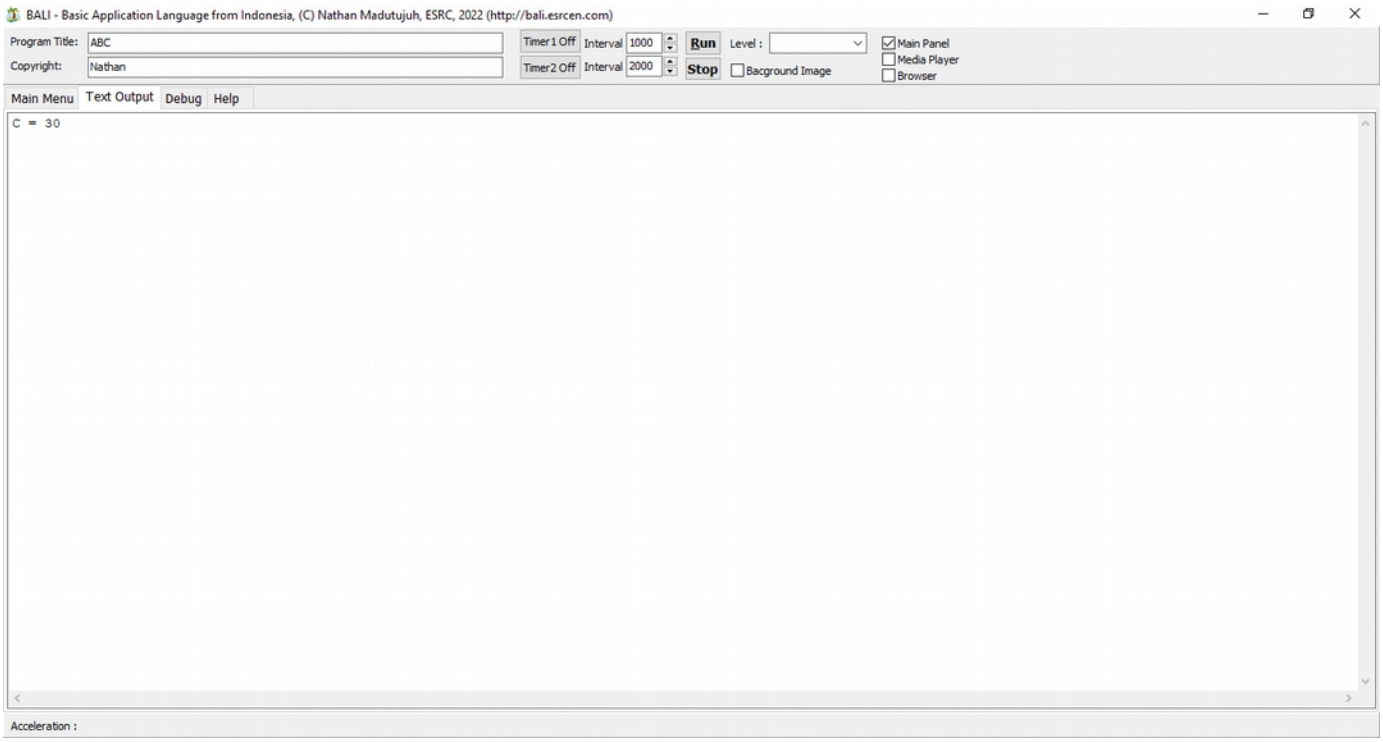
Add 3 Edit buttons A,B and C, and name it accordingly.



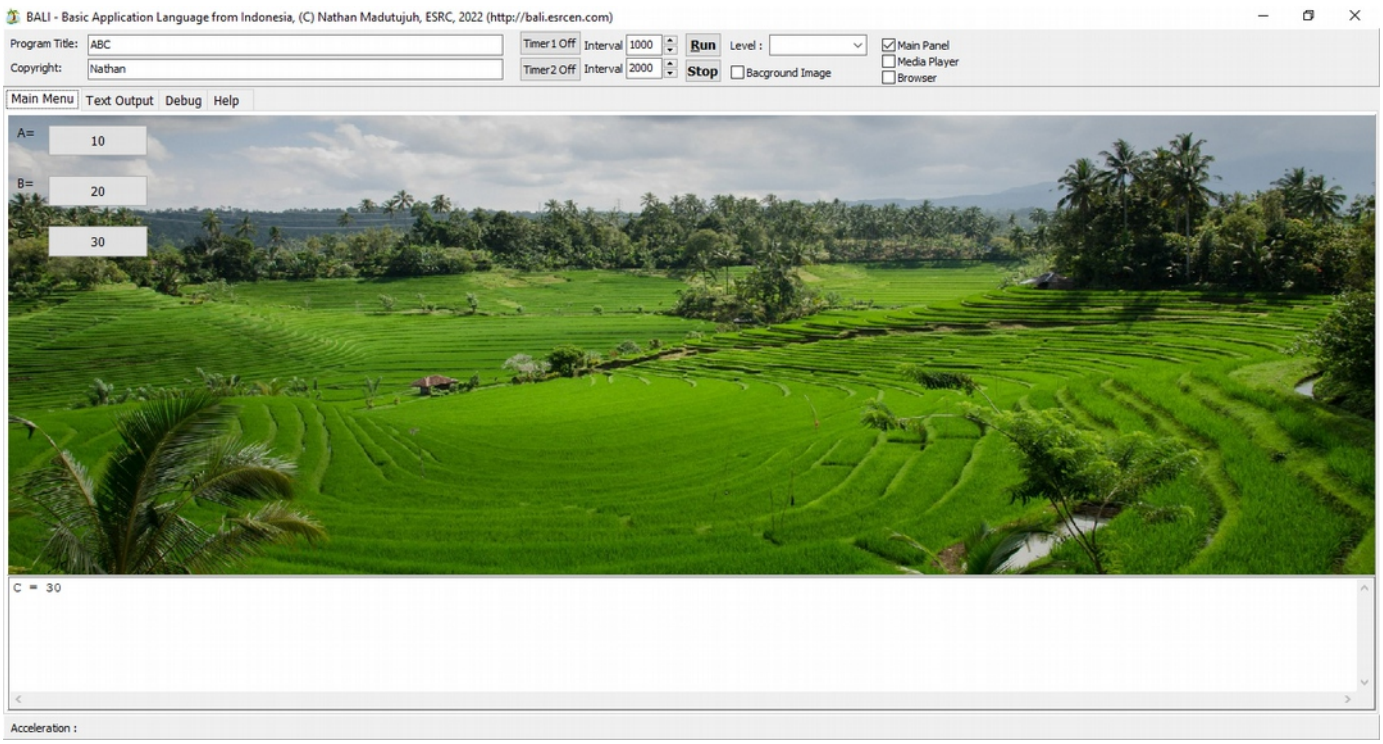
Add code to the main code tab:



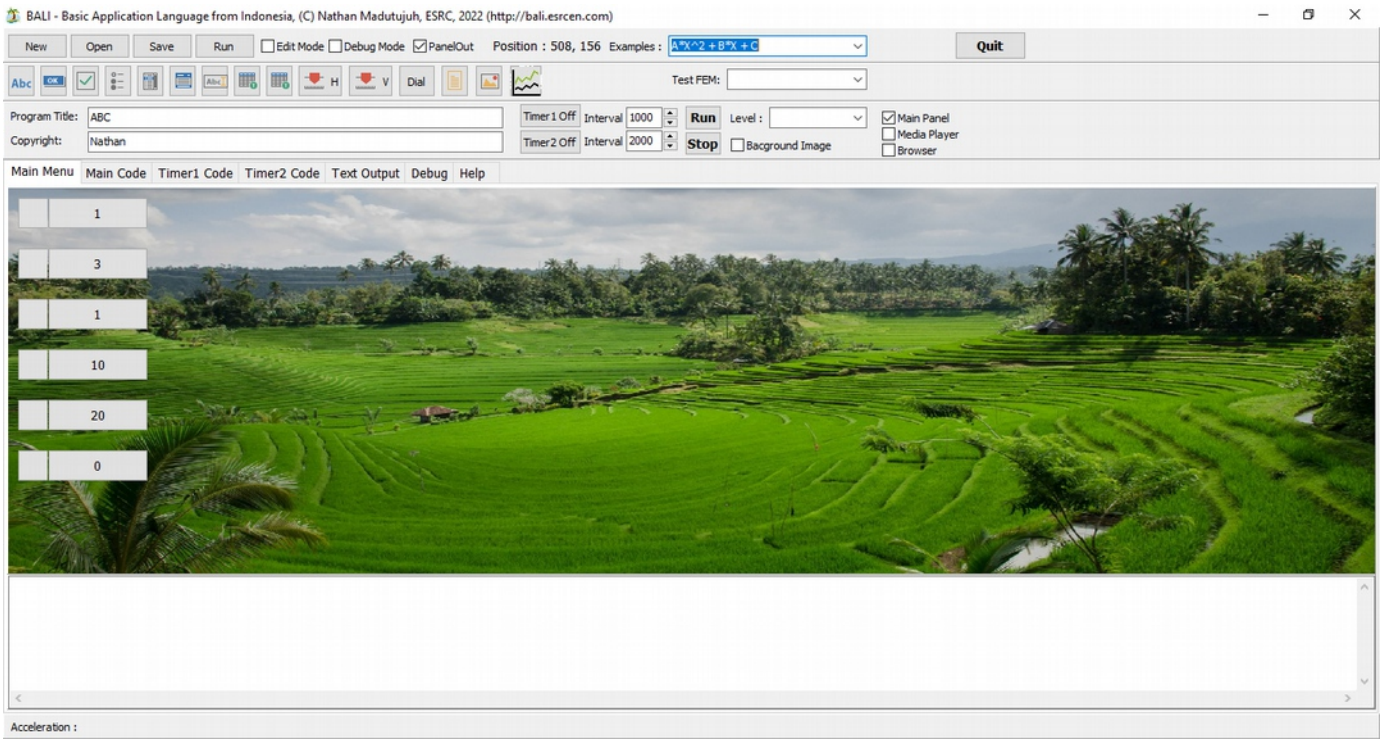
The output will be at text output tab and at visual objects itself as follows:



And at visual object:

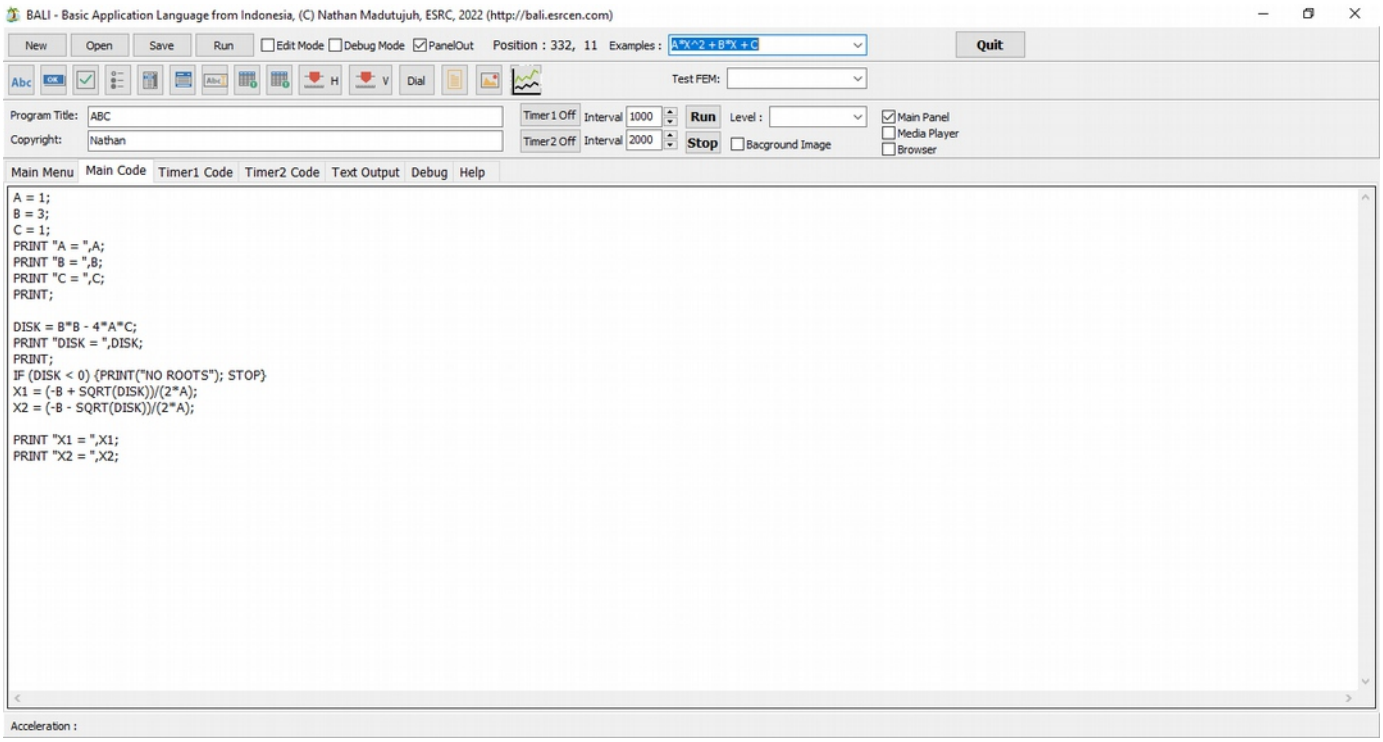


4. ABC Formula

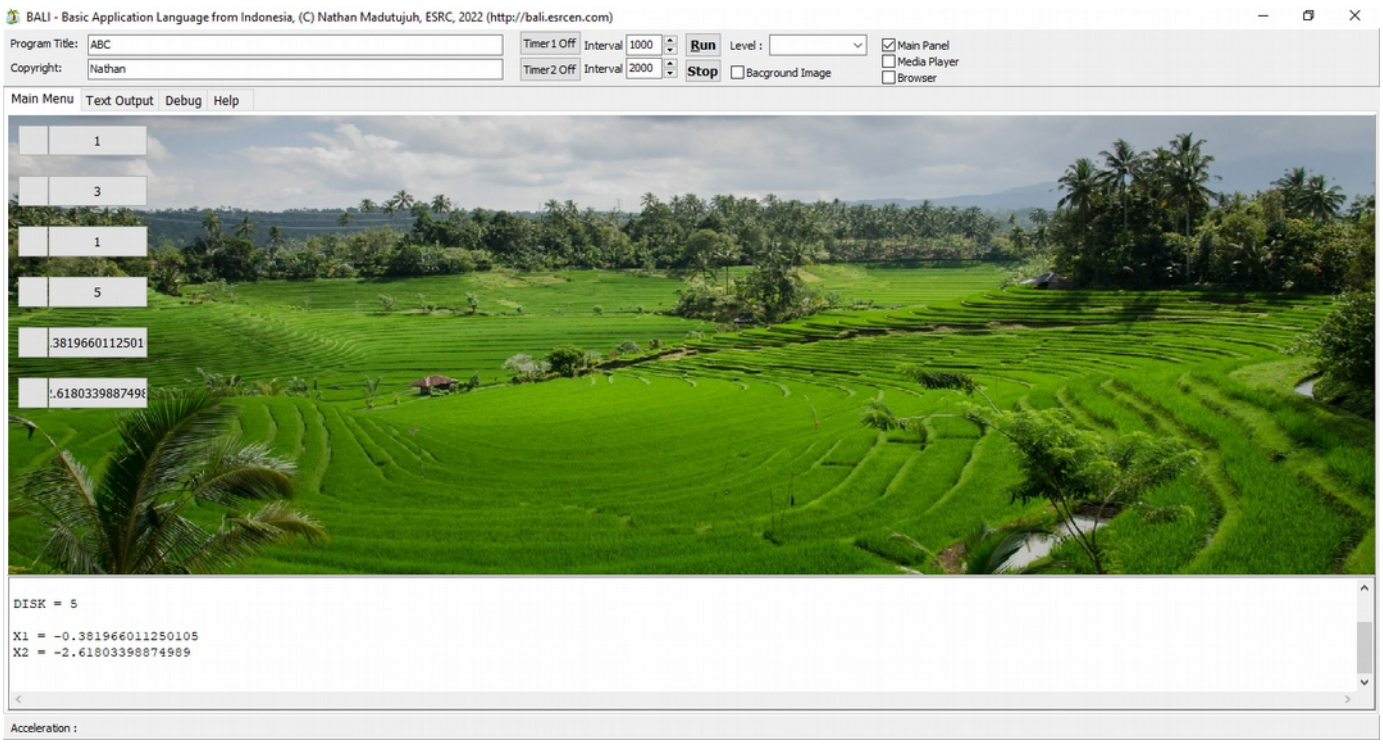


The main code will be:

```
A = 1;
B = 3;
C = 1;
PRINT "A = ",A;
PRINT "B = ",B;
PRINT "C = ",C;
PRINT;

DISK = B*B - 4*A*C;
PRINT "DISK = ",DISK;
PRINT;
IF (DISK < 0) {PRINT("NO ROOTS"); STOP}
X1 = (-B + SQRT(DISK))/(2*A);
X2 = (-B - SQRT(DISK))/(2*A);

PRINT "X1 = ",X1;
PRINT "X2 = ",X2;
```

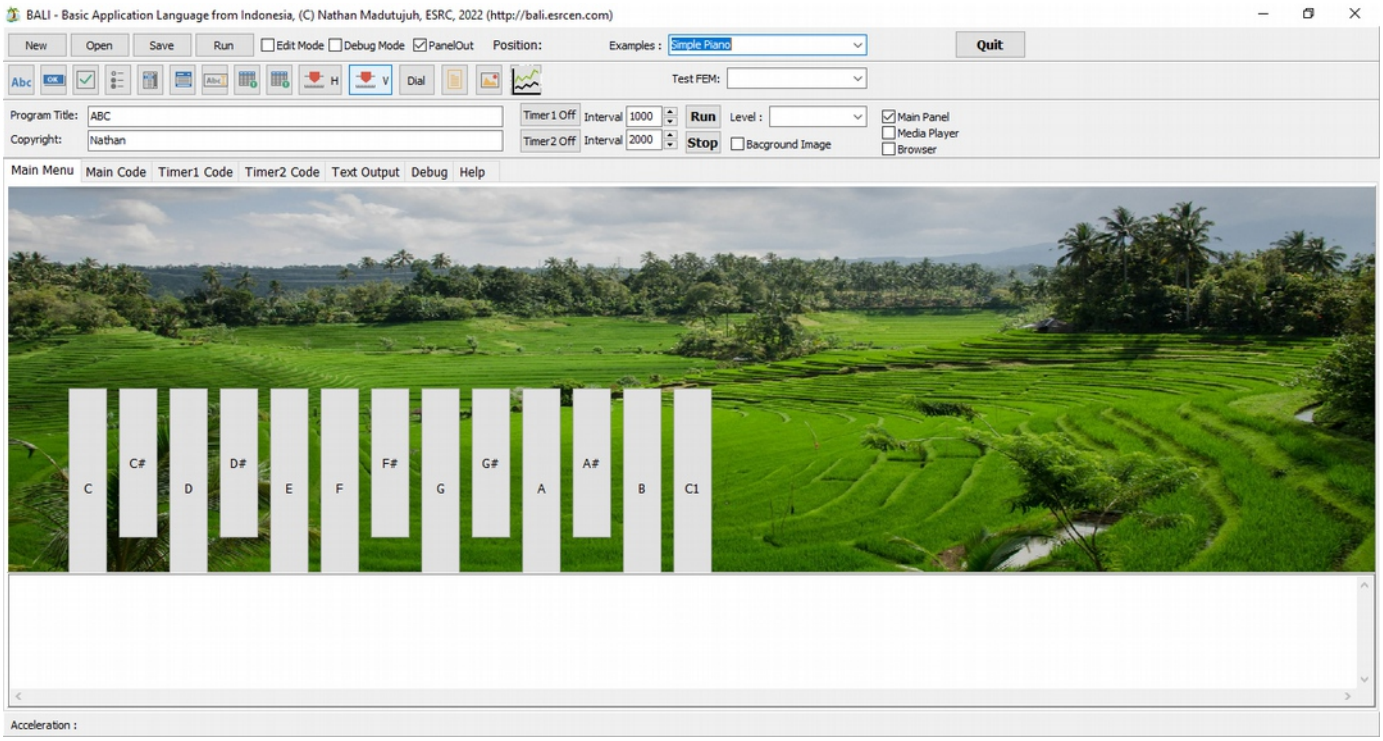


The output will be:



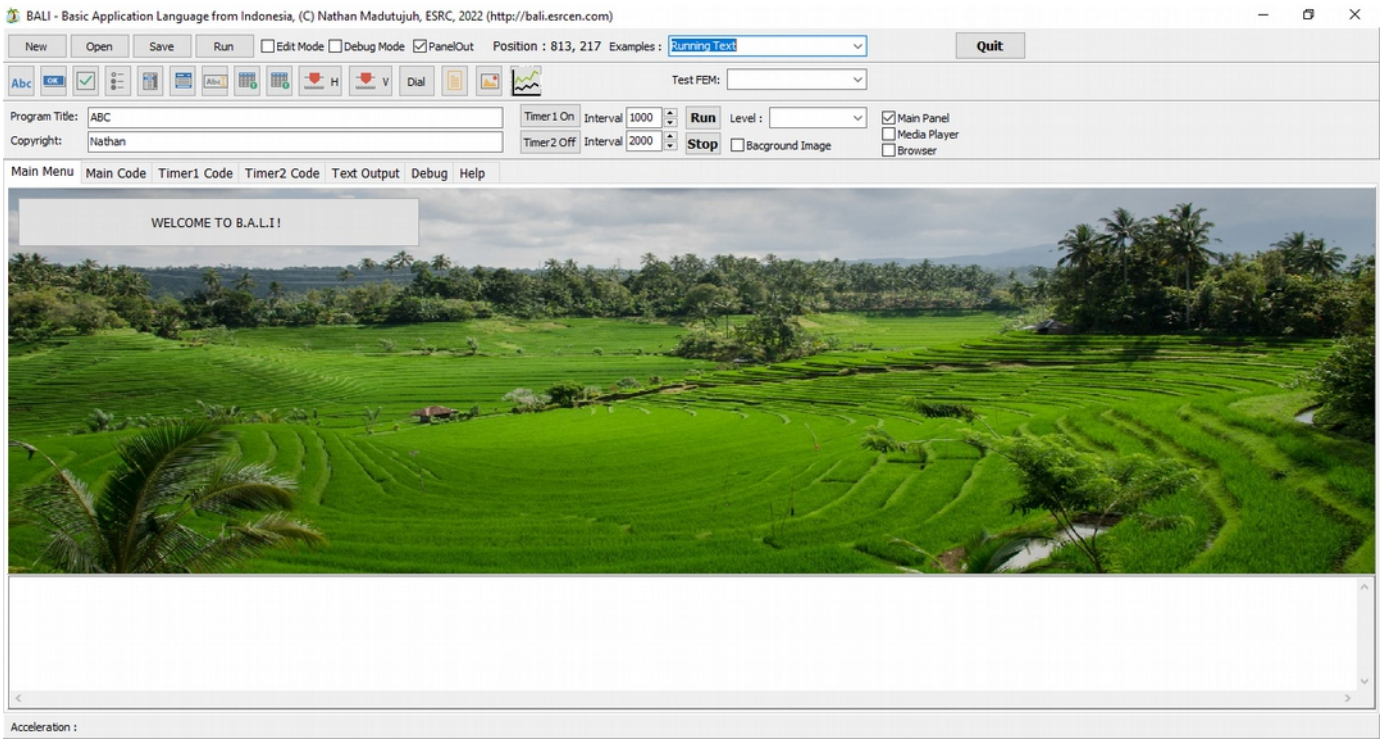
5. Simple Piano

Add several button, rename and select MIDI's instrument, volume, duration and note number for each button according to the note index.



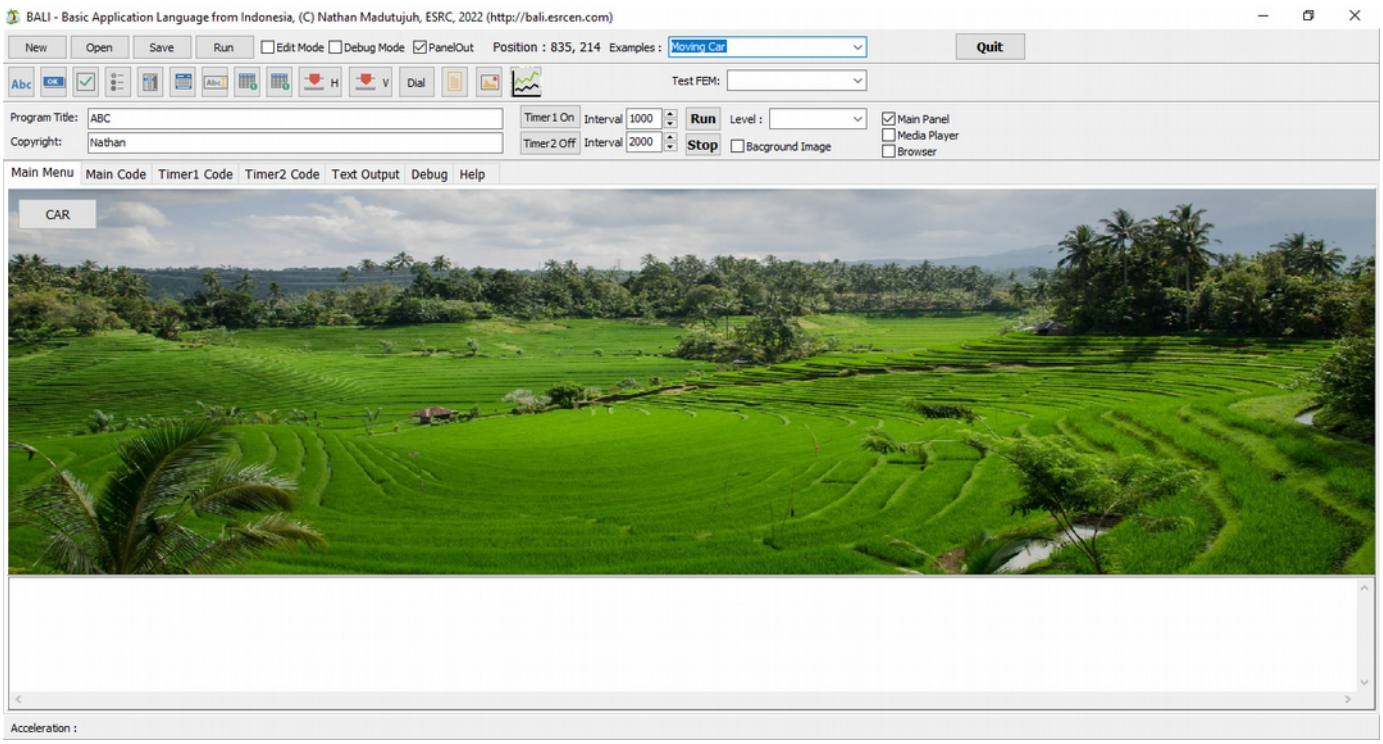
Try the piano, click any button.

6. Text Animation



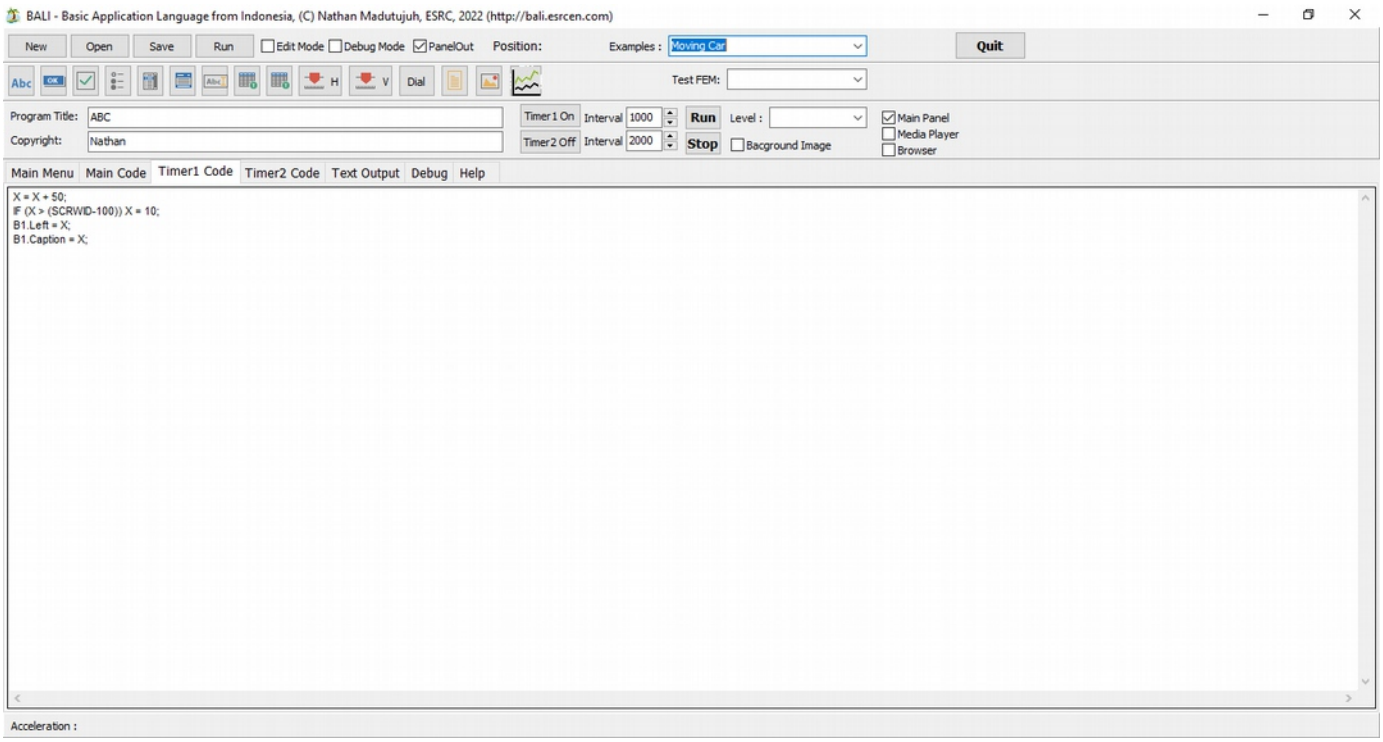
7. Moving Car

Add a button, left position = 10, width = 80, height = 32

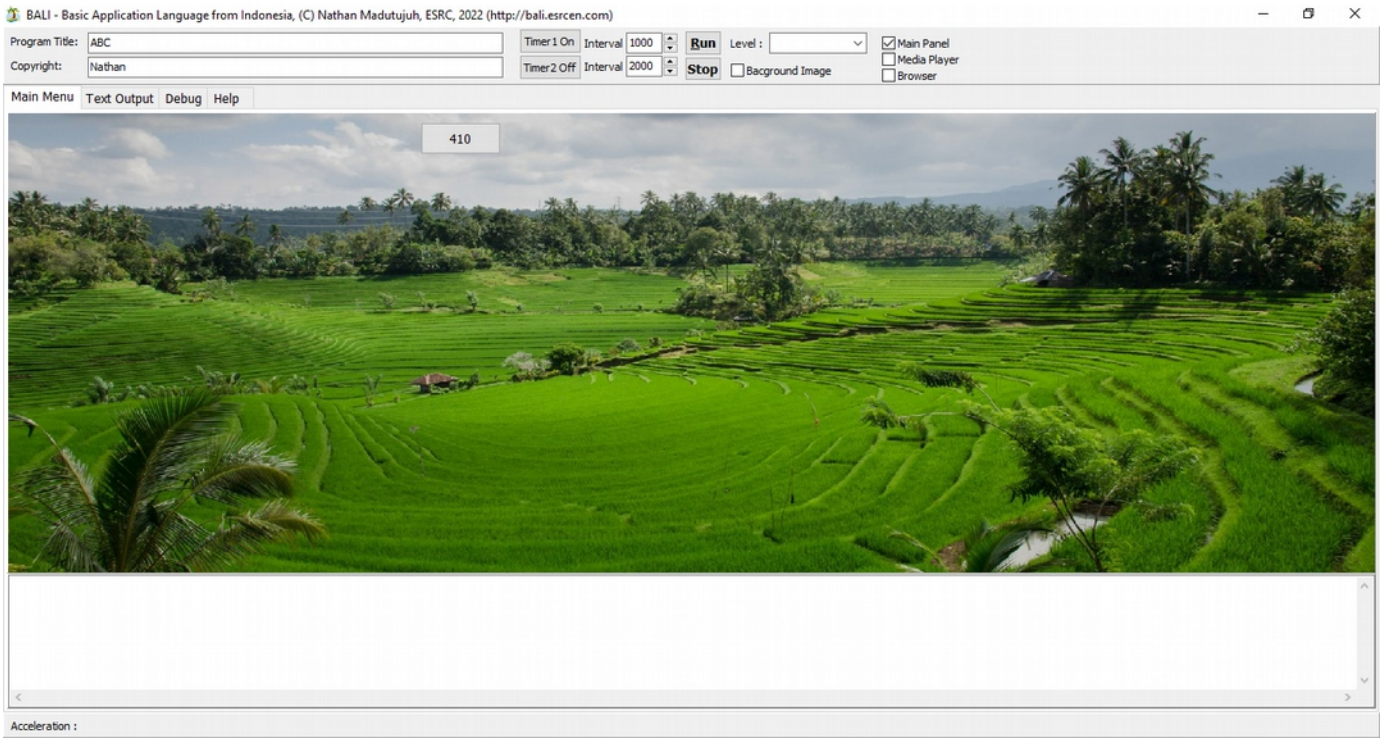


Add at Timer1 code:

```
X = X + 50;
IF (X > (SCRWID-100)) X = 10;
B1.Left = X;
B1.Caption = X;
```



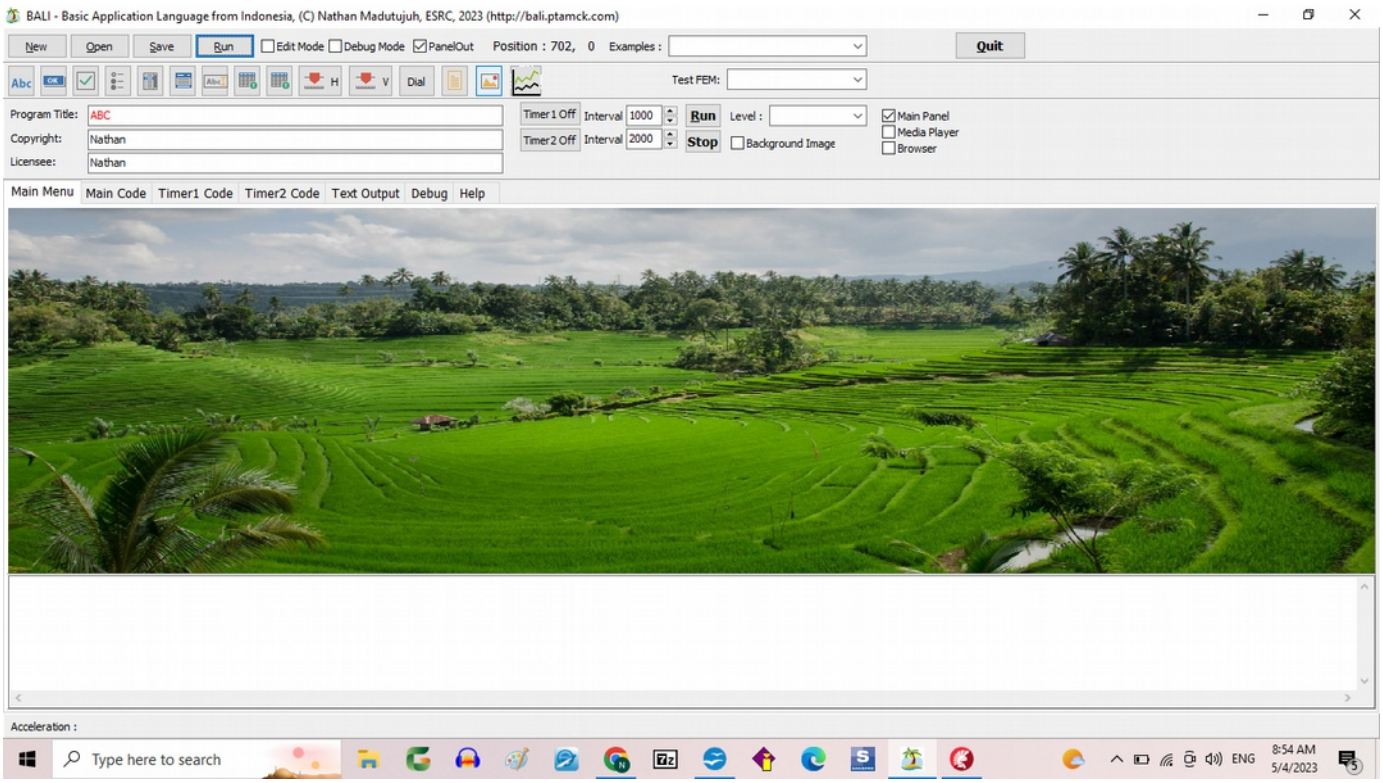
Then click [RUN], Car will move to the right and back.



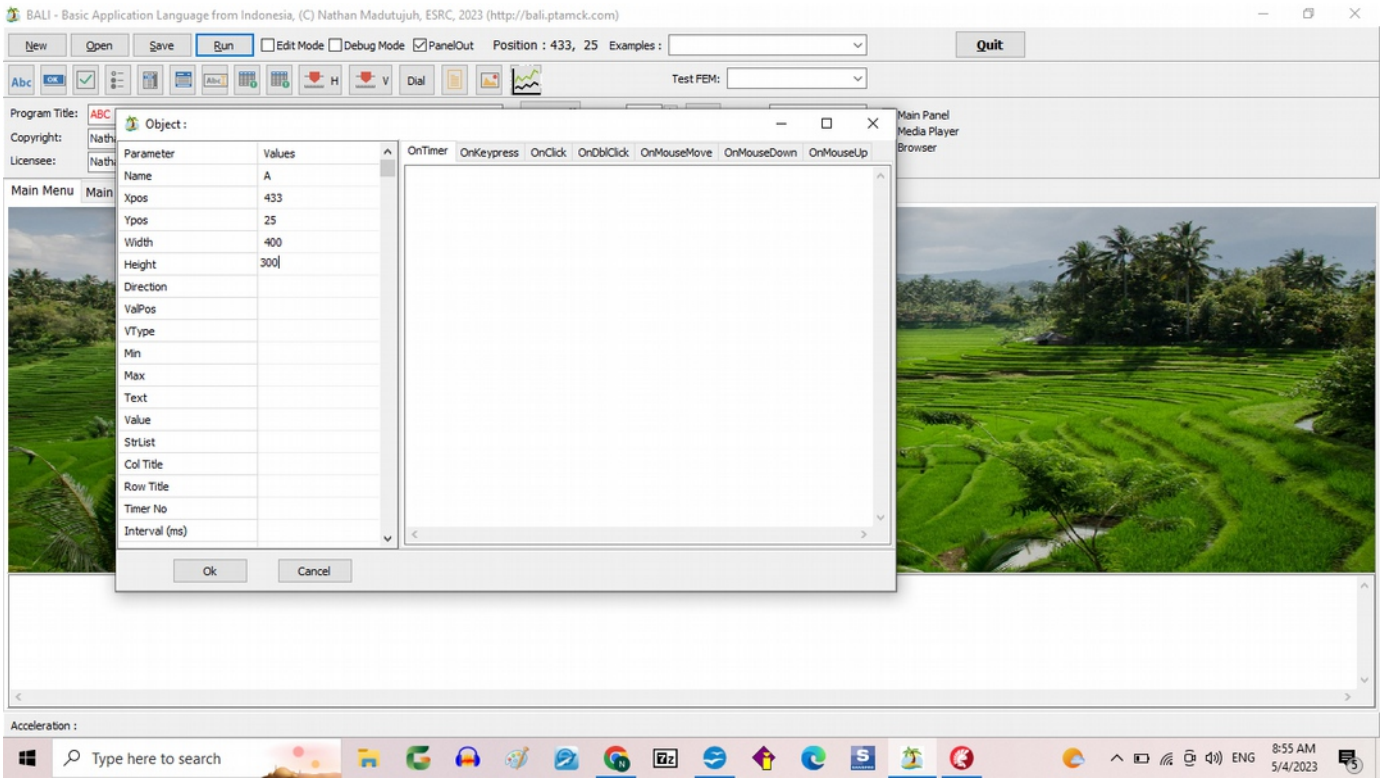
Tips:

Add a second timer for gun to create a simple arcade game.

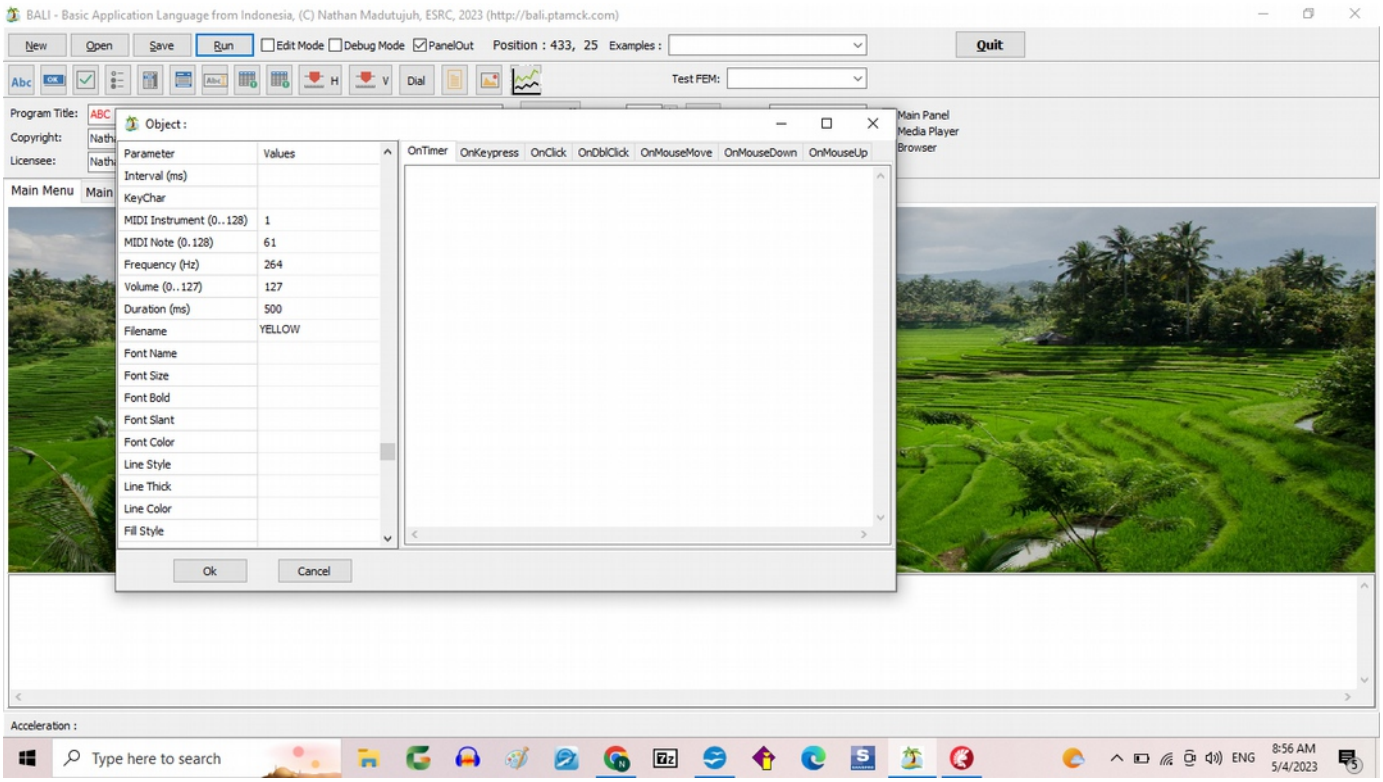
8. **Drawing graphics object with BALI**



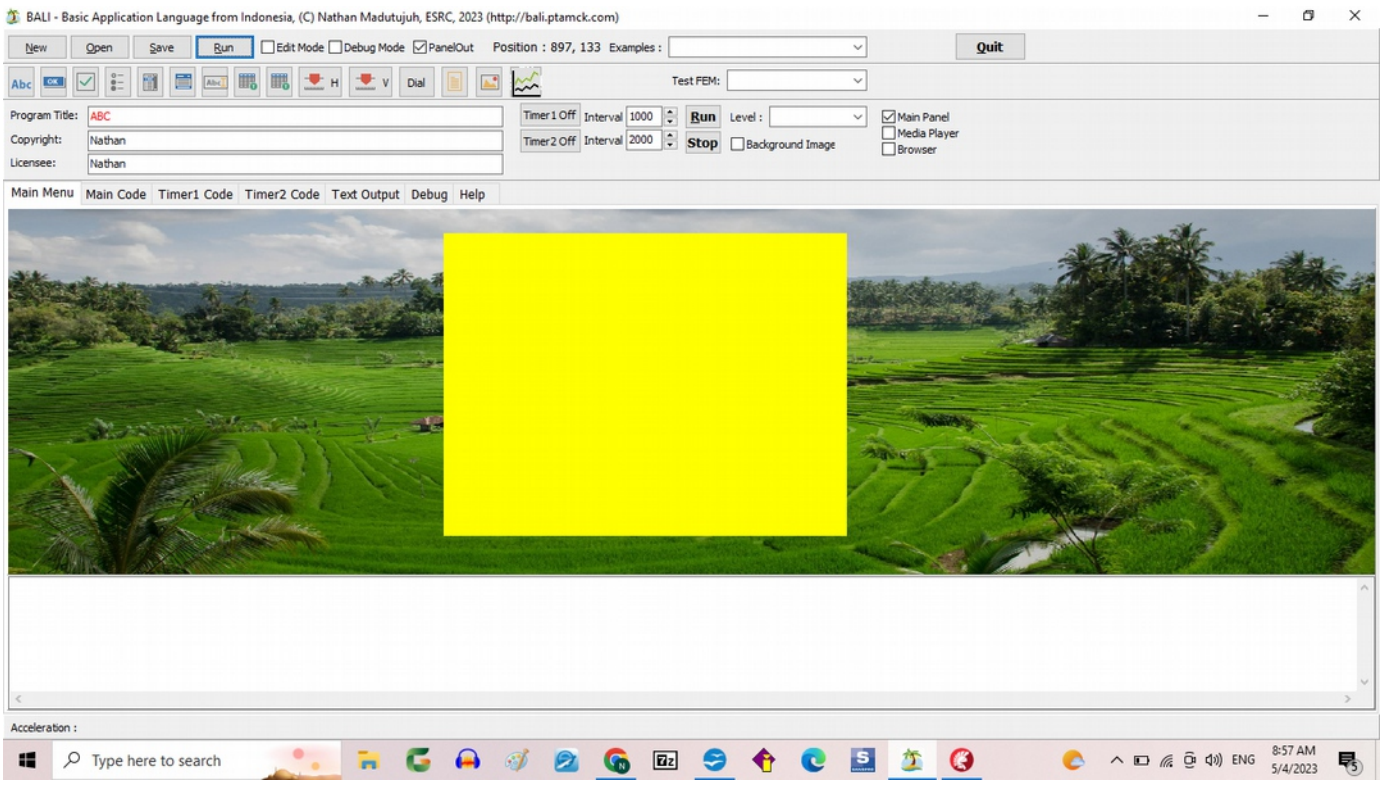
Add an Image Object, name = A, Width = 400, Height = 300



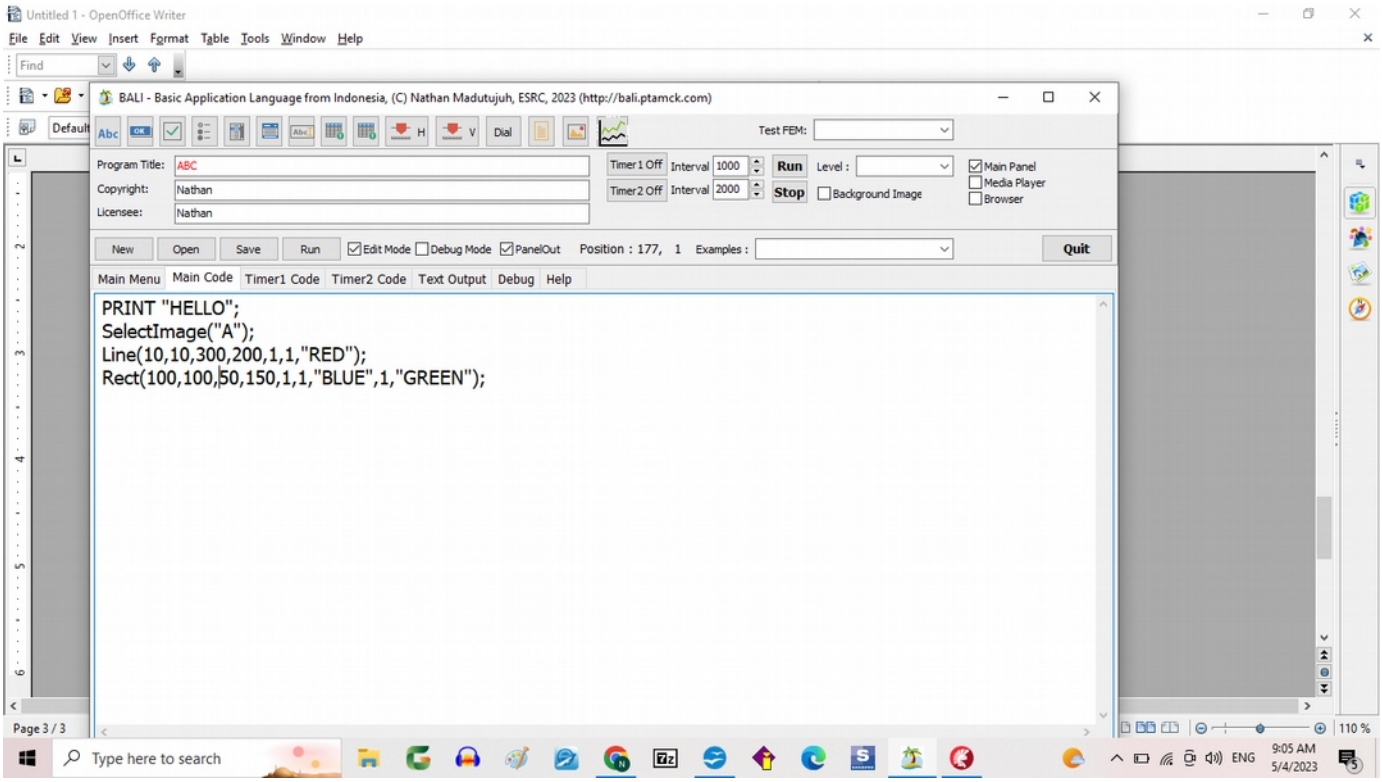
Filename = Yellow or can be entered a file name (test.bmp for example)



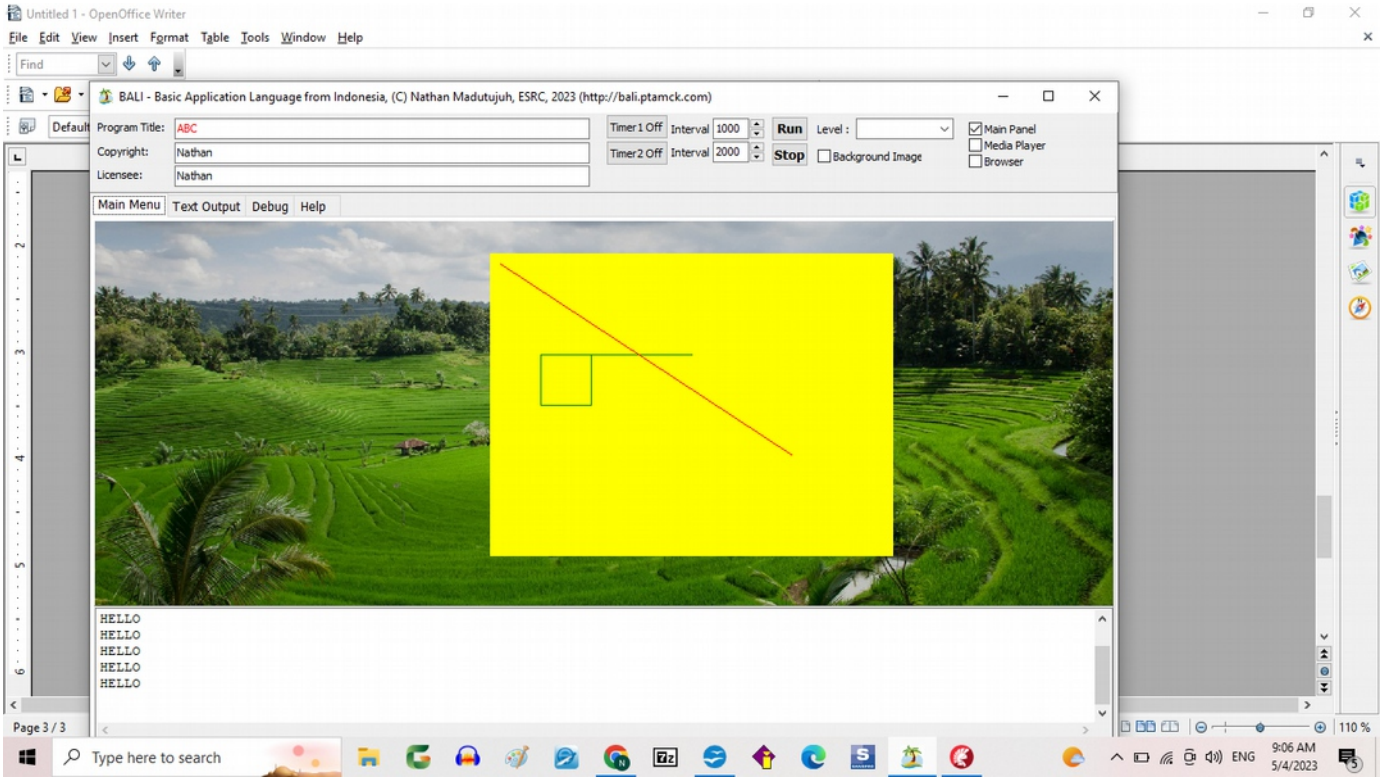
A rectangle yellow box will appear



Enter a code as follows at MainCode Tab



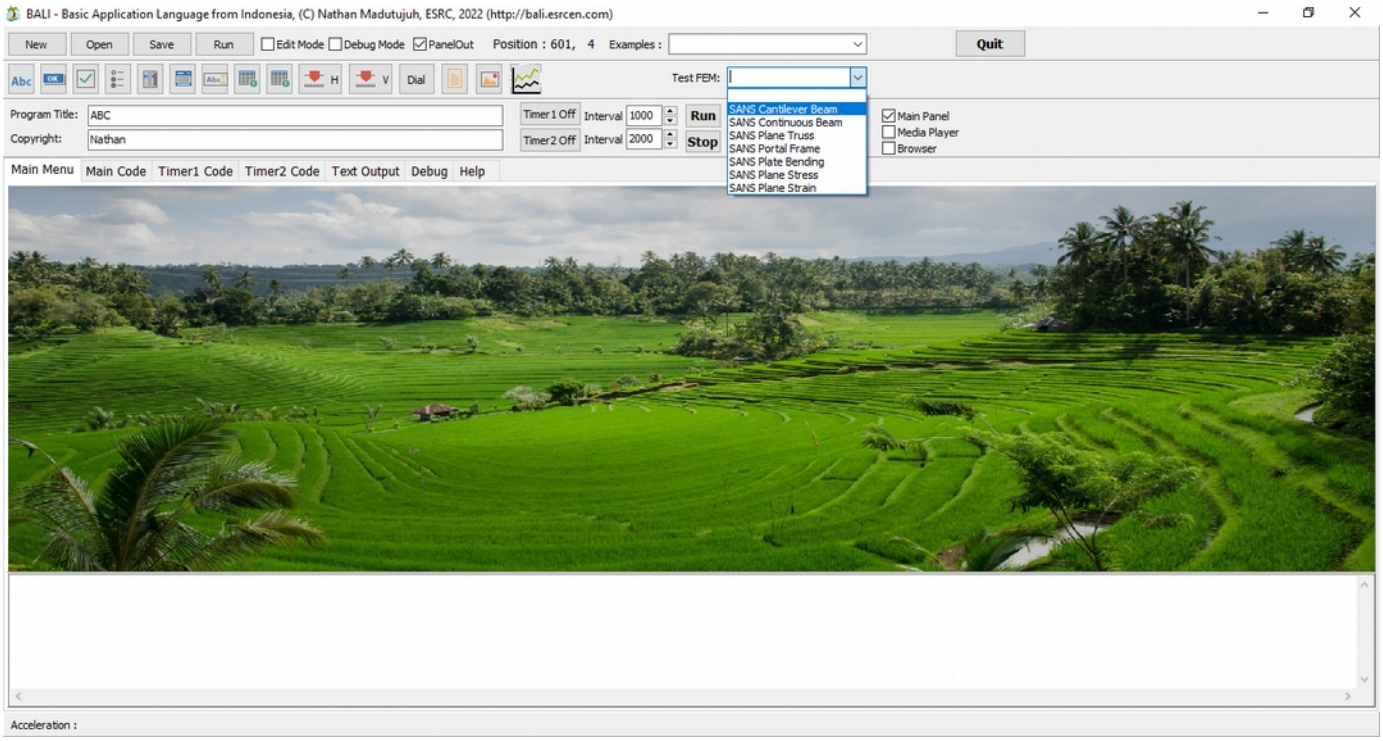
Drawing will appear



Advanced Tutorial:

1. SANSFEM : Simple Cantilever

Select from menu Test FEM: **Cantilever Beam.**

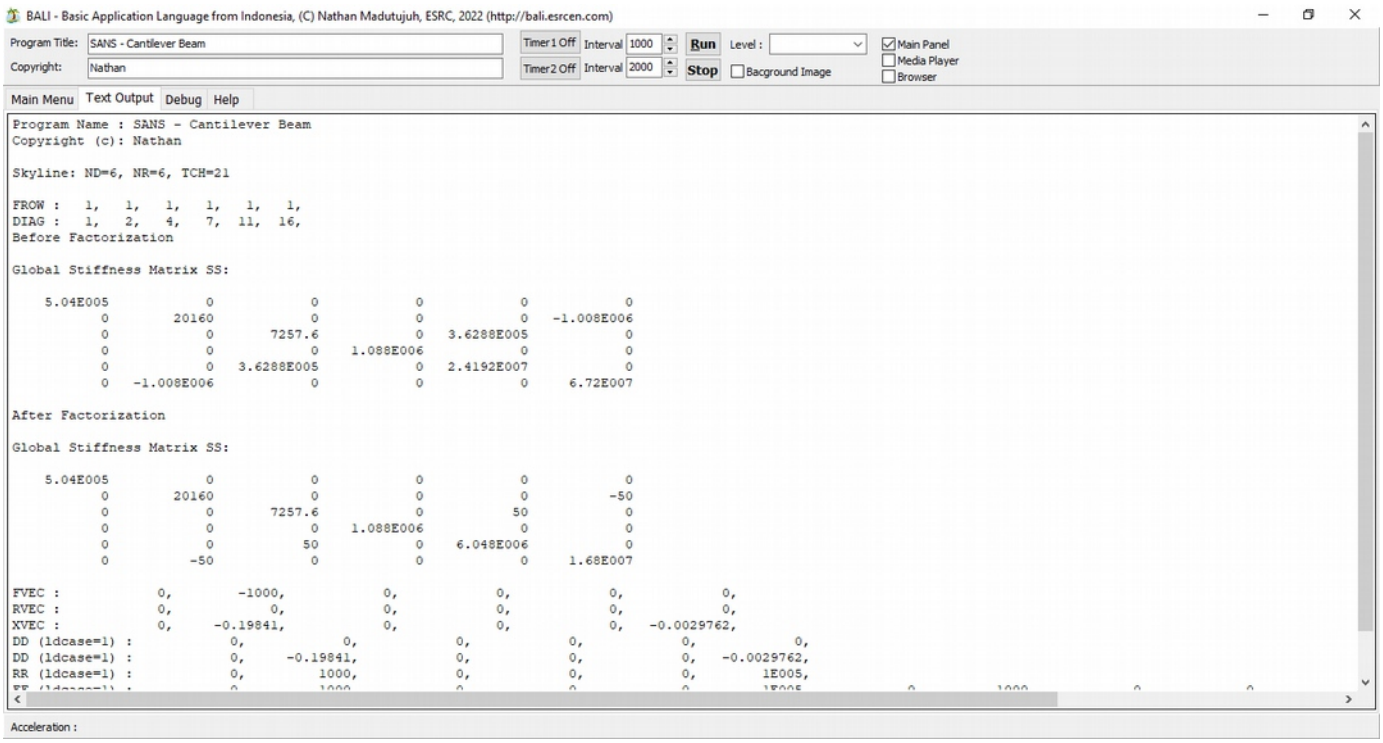


The following text will be added to the Main Code:

```
PRINT_TITLE;
BLANK_ON;

// Initialization:
SANS_CLEARMEM;

//  NLS,NLC,R1,R2,R3,R4,R5,R6, Flag1
SANS_INIT(1,1,0,0,0,0,0,0, 0);

// Material Properties:
// AddProp(matname,sctname : string; Es,G,PR,C,Ywg,Ag,Av,Aw,Ix,Iy,Iz : double;
//                 scttype : integer; d,bw,bf,tf,bb,tb,a : double;
//                 mopt,comp,nb : integer; n,bc,tphc,dc,fc,fy : double) : integer;
SANS_PROP(M1,S1,2.1e5,1e4,0.3,0,0.0024,240,240,240,10880,2880,8000,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0);

// Node Coordinates: id, x,y,z
SANS_NODE(1,0,  0,0);
SANS_NODE(2,100,0,0);

// Support Condition:  id, j,  R1,R2,R3,R4,R5,R6
SANS_SUPPORT(1,1, 1,1,1,1,1,1);

// Truss Element Data : name, n1,n2, mat
// SANS_TRUSS(1,1,2,1);
```

```
// Frame Element Data: name, n1,n2, alpha,rgzf1,rgzf2,mat,rel,rge
SANS_FRAME(1,1,2,0,0,0,1,0,0);

// Load Factor Data: ldcomb, name,  SW,DL,LL,EQX,EQZ,WX,WZ,EPX,EPZ,PS
SANS_LOADCOMB(1,SW,1,0,0,0,0,0,0,0,0,0);

// Joint Load Data: name, ldcase, j, fx,fy,fz,mx,my,mz
SANS_JLOAD(SW,1,2,0,-1000,0,0,0,0);

// Building Skyline Matrix:
SANS_DOF;
SANS_SKYLINE;
SANS_SHOWDOF;
SANS_SHOWFROW;
SANS_SHOWDIAG;

// Compute Global Stiffness:
SANS_STIFF;

// Before Factorization:
PRINT "Before Factorization";
SANS_SHOWSTIFF;

// After Factorization:
PRINT "After Factorization";
SANS_FACTORIZE;
SANS_SHOWSTIFF;

// Compute Load Case : 1
SANS_LOAD(1);
SANS_SHOWFVEC;
SANS_SHOWRVEC;

// Solve for Load Case : 1
SANS_SOLVER(1);
SANS_SHOWXVEC;
SANS_STOREDISP(1);
SANS_COMPUTEFORCE(1);
SANS_STOREREACT(1);
SANS_GETDISP(1,0,1);
SANS_GETDISP(2,0,1);
SANS_GETREACT(1,0,1);
SANS_GETFRAMEF(1,0,1);

SANS_GRAPH(0,1);
```

After click [RUN], the result will be at Text Out tab:



The text output will be:

```
Program Name : SANS - Cantilever Beam
Copyright (c): Nathan

Skyline: ND=6, NR=6, TCH=21

FROW :   1,   1,   1,   1,   1,   1,
DIAG :   1,   2,   4,   7,  11,  16,
Before Factorization

Global Stiffness Matrix SS:

    5.04E005            0            0            0            0            0
           0        20160            0            0            0   -1.008E006
           0            0       7257.6            0   3.6288E005            0
           0            0            0    1.088E006            0            0
           0            0   3.6288E005            0   2.4192E007            0
           0   -1.008E006            0            0            0     6.72E007

After Factorization

Global Stiffness Matrix SS:

    5.04E005            0            0            0            0            0
           0        20160            0            0            0          -50
           0            0       7257.6            0           50            0
           0            0            0    1.088E006            0            0
           0            0           50            0    6.048E006            0
           0          -50            0            0            0     1.68E007

FVEC :            0,        -1000,            0,            0,            0,            0,
RVEC :            0,            0,            0,            0,            0,            0,
XVEC :            0,     -0.19841,            0,            0,            0,   -0.0029762,
DD (ldcase=1) :            0,            0,            0,            0,            0,            0,
DD (ldcase=1) :            0,     -0.19841,            0,            0,            0,   -0.0029762,
RR (ldcase=1) :            0,         1000,            0,            0,            0,        1E005,
FF (ldcase=1) :            0,         1000,            0,            0,            0,        1E005,
0,        -1000,            0,            0,            0,  1.1099E-011,            0,       -50000,
```

2. SANSFEM: Simple Beam

3. SANSFEM: Roof Truss

4. SANSFEM: Portal Frame

Advanced Features:

1. String functions

```
Trim(s)          // remove pre and post spaces
UpperCase(s)     // change all characters to uppercase
LowerCase(s)     // change all characters to lowercase
Copy(s,p,n)      // copy a substring from location p, n characters
IntToStr(n)      // convert integer n to string
NumToStr(x)      // convert numeric x to string
TimeToStr(t)     // convert time t to string
DateToStr(d)     // convert date d to string
Length(s)        // length of a string
Pos(s1,s)        // find a string s1 inside s
Val(s)           // value of a string
StrToInt(s)      // string to integer
StrToNum(s)      // string to numeric
StrToTime(s)     // string to time
StrToDate(s)     // string to date
```

2. Math functions

(n = integer, x = floating number)

Integer Functions:

```
Round(x)            // Round to nearest integer
Trunc(x)            // Truncated to nearest lower integer
Int(x)              // Truncated to integer part
Ceil(x)             // Get nearest higher integer
Floor(x)            // Get nearest lower integer
Even(n)             // 0 if not even number, 1 if even number
Odd(n)              // 0 if not odd  number, 1 if odd  number
Fact(n)             // factorial : 1*2*3*n-1*n
Inc(n)              // n+1
Dec(n)              // n-1
RandomRange(n1)     // 0..n1-1
RandomRange(n1,n2)  // n1..n2
```

Float Functions :

```
Random()            // 0..1
Random(n1)          // 0..n1-1
Random(n1,n2)       // n1..n2
Abs(x)              // absolute value of x
Sign(x)             // sign of x : +1 if > 0, 0 if zero, -1 if < 0
Deg(x)              // Degree of a radiant angle
Rad(x)              // Radiant of a degree angle
Sqrt(x)             // square root of x
Sqr(x)              // x^2
Power(x,y)          // x^y
Exp(x)              // e^x
Log(x)              // Ln(x)/Ln(10)
Log(a,b)            // Ln(a)/Ln(b)
LogN(a,b)           // Ln(a)/Ln(b)
Log2(x)             // Log2(x)
Log10(x)            // Log10(x)
Ln(x)               // Ln(x)
Max(x1,x2,...)      // max of (x1,x2,...)
Min(x1,x2,...)      // min of (x1,x2,...)
Length(x1,x2,...)   // Sqrt(x1^2 + x2^2 + ....)
Avg(x1,x2,...)      // (x1+x2+...)/n
Sum(x1,x2,...)      // x1 + x2 + ...
SumSqr(x1,x2,...)   // x1^2 + x2^2 + ....
SRSS(x1.x2,...)     // Sqrt(x1^2 + x2^2 + ....)
CQC(x1,x2,...)      // CQC(x1,x2,...)
```

Linear Interpolation:

```
LinearInterp(x,x1,y1,x2,y2);  // Linear interpolation of y using x value
```

Trigonometric functions:

```
Sin(x)              // Sine of x, radian
Cos(x)              // Cosine of x, radian
```

```
Tan(x)          // Tangent of x, radian
Sec(x)          // Secant of x, radian
Cosec(x)        // Cosecant of x, radian
CoTan(x)        // CoTangent of x, radian
SinD(x)         // Sine of x, degree
CosD(x)         // Cosine of x, degree
TanD(x)         // Tangent of x, degree
SecD(x)         // Secant of x, degree
CosecD(x)       // Cosecant of x, degree
CoTanD(x)       // CoTangent of x, degree
Asin(x)         // ArcSine of x, radian
Acos(x)         // ArcCosine of x, radian
Atan(x)         // ArcTangent of x, radian
Asec(x)         // ArcSecant of x, radian
Acosec(x)       // ArcCoSecant of x, radian
Acotan(x)       // ArcCoTangent of x, radian
SinH(x)         // Sine Hyperbolic of x, radian
CosH(x)         // Cosine Hyperbolic of x, radian
TanH(x)         // Tangent Hyperbolic of x, radian
SecH(x)         // Secant Hyperbolic of x, radian
CosecH(x)       // Cosecant Hyperbolic of x, radian
CoTanH(x)       // Cotangent Hyperbolic of x, radian
AsinH(x)        // Arc Sine Hyperbolic of x, radian
AcosH(x)        // Arc Cos Hyperbolic of x, radian
AtanH(x)        // Arc Tangent Hyperbolic of x, radian
AsecH(x)        // Arc Secant Hyperbolic of x, radian
AcosecH(x)      // Arc Cosecant Hyperbolic of x, radian
AcoTanH(x)      // Arc CoTangent Hyperbolic of x, radian
```

Calculus :

Vector :

```
VecLen(a,b,c,...)   // Length    : sqrt(a^2 + b^2 + c^2 + ….)
VecSum(a,b,c,...)   // Sum       : a + b + c + ...
VecAvg              // Average   : (a + b + c + ….)/n
VecSumSqr           // SumSqr    : a^2 + b^2 + c^2 + ….
VecSRSS             // SRSS      : sqrt(a^2 + b^2 + c^2 + ….)
```

Matrix :

```
MATRIX_NULL(A)          // Matrix, all 0
MATRIX_UNIT(A)          // Matrix, all 1
MATRIX_RANDOM(A)        // Matrix, all random numbers
MATRIX_COL(A,n)         // Column n of matrix A
MATRIX_ROW(A)           // Row n of matrix A
MATRIX_TRIDIAG(A)       // Trdiagonal Matrix A
MATRIX_TRANSPOSE(A)     // Transpose of A
MATRIX_DETERMINANT(A)   // Determinant of A
MATRIX_INVERSE(A)       // Inverse of A
MATRIX_FACTORIZE(A)     // Factorize of A = L.D.L^T
MATRIX_SOLVER(A,X,Y)    // Solve A.x = y
```

3. File functions

```
OpenInpText(fname)          // Open input text file
CloseInpText(fname)         // Close input text file
Readln(v1,v2,v3,...)        // Read a line from input text file

OpenOutText(fname)          // Open output text file
CloseOutText(fname)         // Close output text file
Writeln(v1,v2,v3,...)       // Write several variables
```

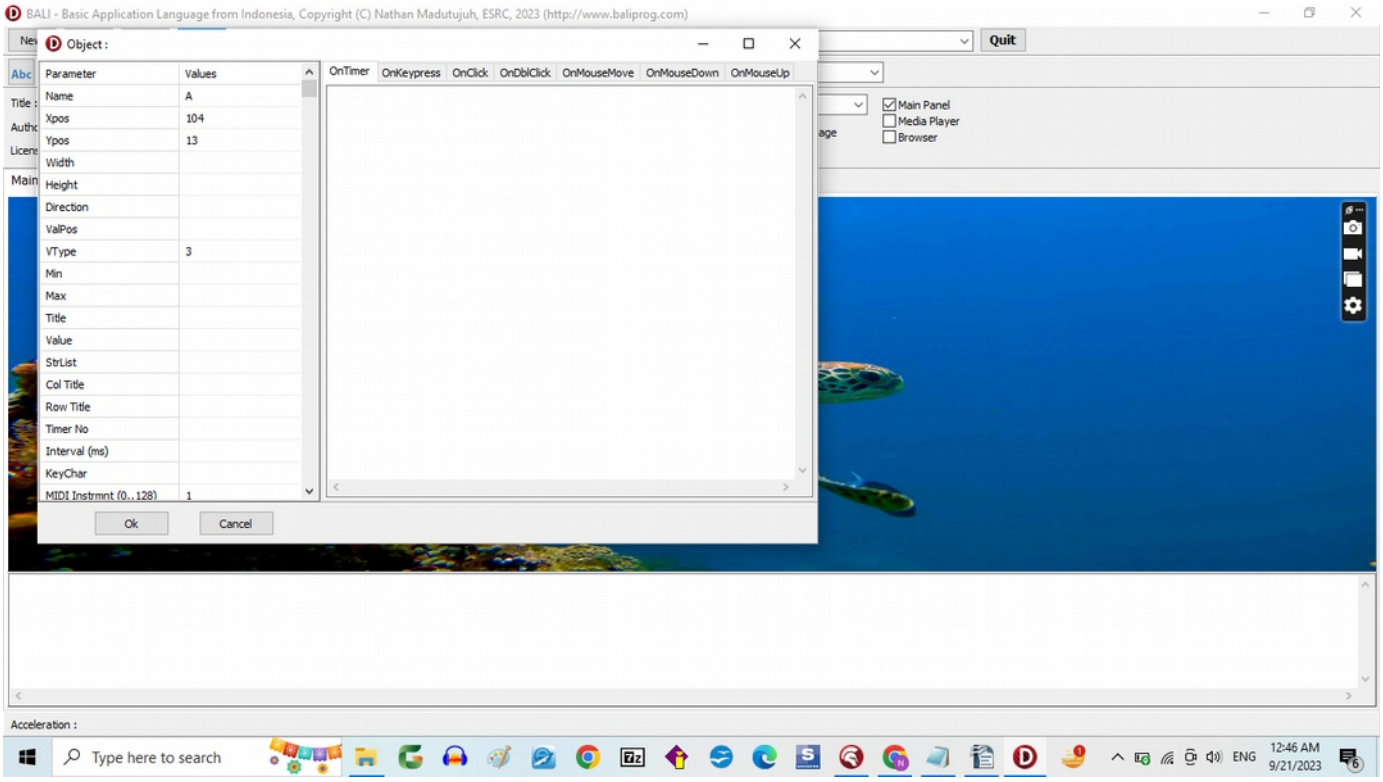
4. Graphics functions

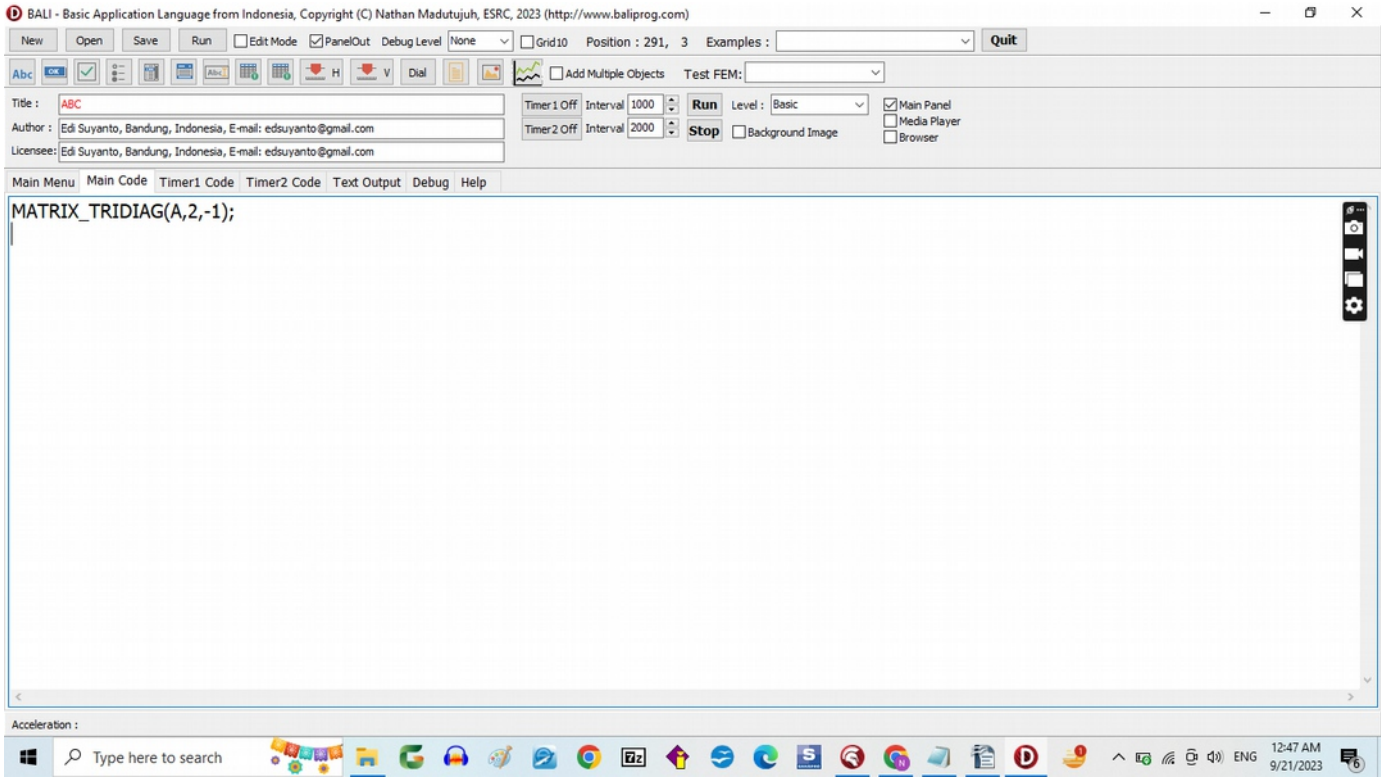
To draw a graphic, first select Target Image, then use any graphics command to draw on the target image. Using this method, we can draw to several images easily.

```
SCRWID                          // Screen Width in pixels
SCRHGT                          // Screen Height in pixels

SelectImage(objname)            // Select Image from an object
PenMode(pm)                     // Set Pen Mode
Color(color)                    // Set color(color)
Sound(filename,duration,pitch)  // Play a wav file

Text(x,y,agl,fsize,fcolor,txt)                          // Draw a string with angle agl, size fsize
Line(x1,y1,x2,y2,thick,style,color)                     // Draw a line from x1,y1 to x2,y2
Rect(x1,y1,x2,y2,thick,style,color,fillstyle,fillcolor) // Draw a rectangle from x1,y1 to x2,y2
Circle(x,y,r[,thick,style,color,fillstyle,fillcolor)    // Draw a circle using x,y,r
Ellipse(x,y,rx,ry,thick,style,color,fillstyle,fillcolor) // Draw an ellipse using x,y,rx,ry
Polygon(x,y,r,n,startagl,thick,style,color,fillstyle,fillcolor)  // Draw a n-sides polygon
```

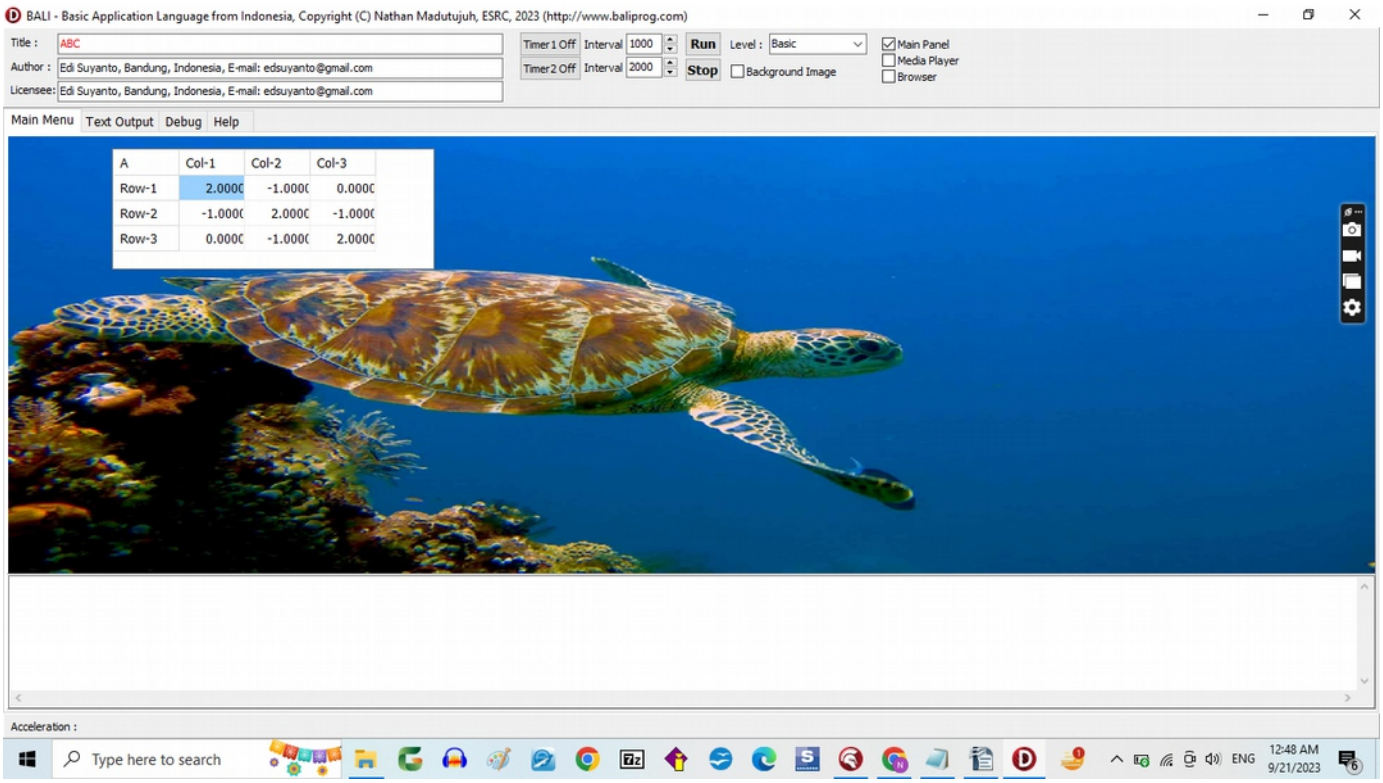
5. Vector and Matrix

Add a 3x3 matrix, type = double, name = A
Add a 3x1 vector, type = double, name = X
Add a 3x1 vector, type = double, name = B
Set A as a tridiagonal matrix, Dii = 2.0, Dij = -1.0;
Set B[1] = 1.0;
Solve for X

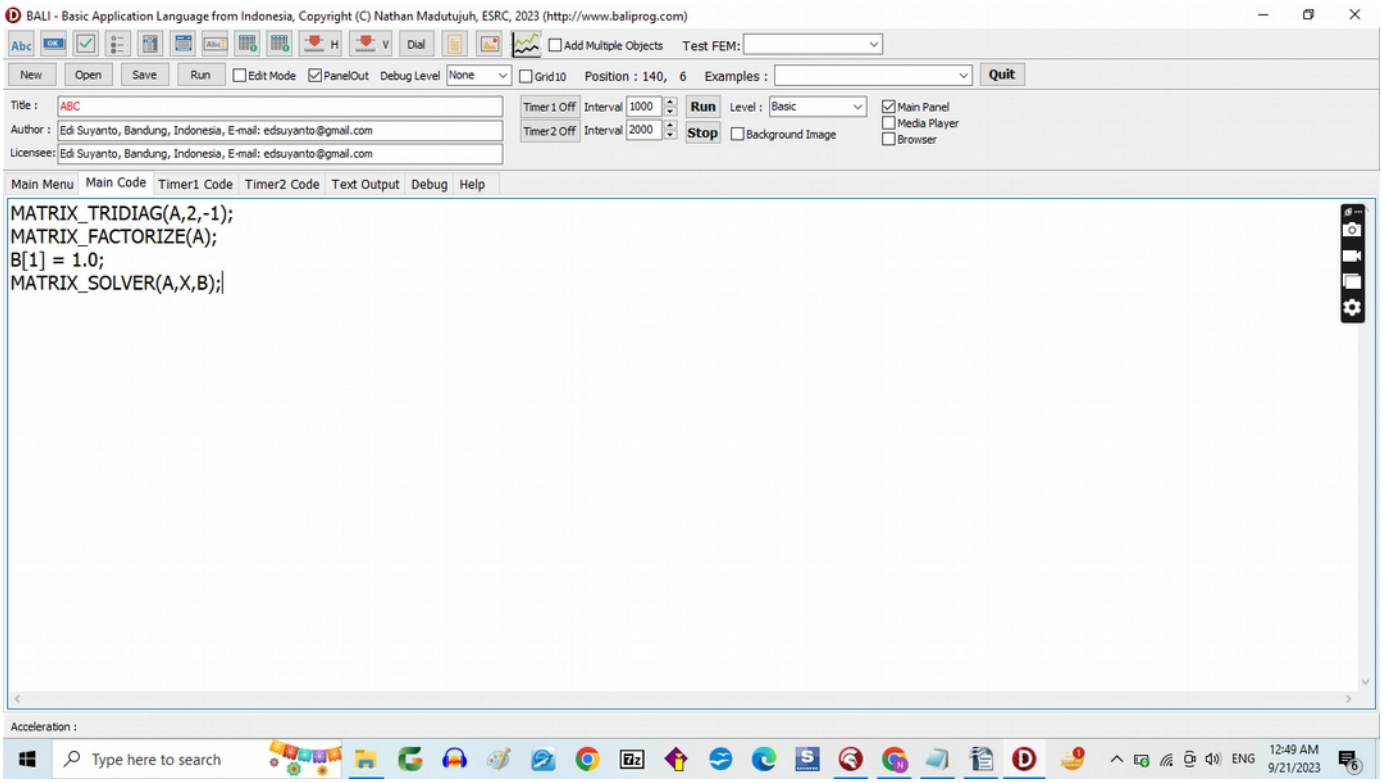




A Tridiagonal matrix will appear:



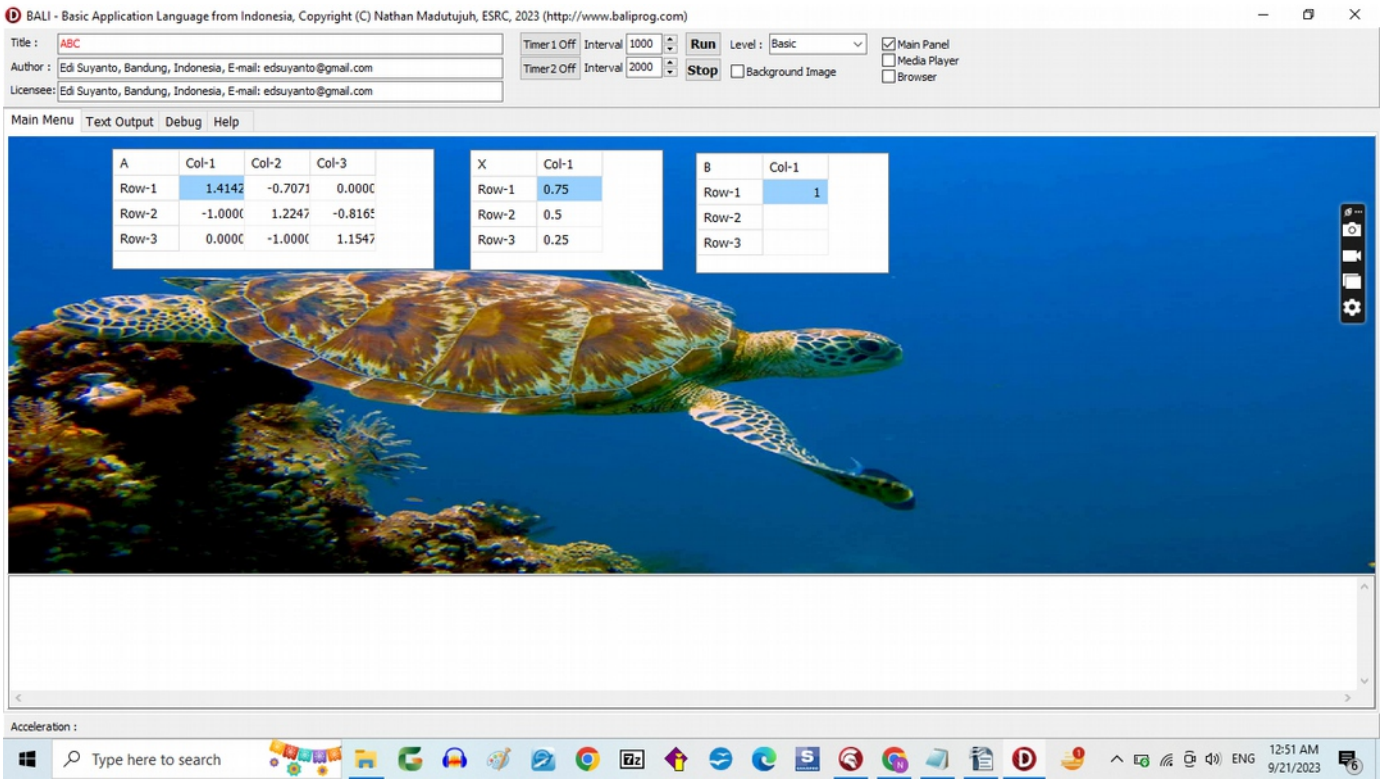
Factorize the A matrix into Cholesky L*U Format and Solve for X:



B = {1.0, 0, 0}

The Solver will use Cholesky Factorized Matrix A to solve the linear equation.

The result is :



X = {0.75, 0.5, 0.25}

Other Matrix commands:

MATRIX_INVERSE(A); → To get the inverse of matrix A

BALI Language Syntax Reference

1. Identifier

Valid chars: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789_
.

Examples:	_aBc123	a variable
	A.Color	an object parameter

2. Constants

String	: "abc123"
Integer	: 1,12,-30
Float	: -123.456, -123.456E-123
Special	: Pi, Euler

3. Variable

Variable will be automatically defined globally when first found
Object Name is a variable
Visual Object text will be updated automatically due to the related variable change

4. Statements and Blocks

Statements are consisted of single line statements or multiple statements surrounded by {} named a block. A block will be treated logically as a single statement. Each statement can be ended with ; (optional).

```
s1
{s1; s2; s3}
```

5. Operators and Keywords

Assignment	: =
Numeric	: +,-,*,/,^
Unary	: ++,--,+=,-=,/=,*=,^=
Logic	: !,NAND AND NOR XOR OR NOT
Equality	: <,>,==,<>,!=,<=,>=
Keywords	: IF, ELSE, FOR, TO, STEP, DO, WHILE, REPEAT, UNTIL

6. Expressions

expression : any combination of parentheses, identifiers and operators

Numeric expression:

-exp
!exp
exp2 + exp2
var1 += exp1
var1 -= exp1
var1 *= exp1

```
var1 /= exp1
var1 ^= exp1
exp
sin(exp)
exp1+exp2-exp3
(exp)
```

Logical expression:

(a and b must be surrounded by () if more than one entities)
(a and b can be logical expression or numeric expression or mixed)

```
!a
a != b
a <> b
a > b
a < b
a >= b
a <= b
```

7. Control Structures

Control structures can be used to direct the flow of code execution, logical decision and repetition. Single line control structures contain everything in one line, while Multi-line control structures can have multiple lines for each block.

a. Single line control structures:

```
IF () s1;
IF () s1 ELSE s2;
LOOP (i,i1,i2,n) s1;
FOR i = j TO k STEP n DO s1;
WHILE (e) DO s1;
REPEAT s1 UNTIL (e);
```

b. Multi-line control structures

```
IF (a) {
  ...
}

IF (a) {
  ...
} ELSE {
  ...
}

SWITCH (v) {
  a : {s1};
  b : {s2};
  c : {s2};
}

LOOP (i,i1,i2,n) {
}
```

```
FOR i = j TO k STEP n DO {
}

WHILE (e) DO {
}

REPEAT
  ...
UNTIL (e)
```

BALI Functions List

1. MIDI and sound Functions

(Dura in milliseconds)

```
Delay(Dura)
Beep(Freq,Dura)
PlayMidi(Instrum,Note,Volume,Dura)
PlayABC('ABCDEFGA')                 // Instrum=1=Piano, Volume=100, Dura=250
Play123('1234567i')                 // Instrum=1=Piano, Volume=100, Dura=250
NoteOn(Note,Volume)
NoteOff(Note,Volume)
```

2. Input/Output/Keyboard Functions and Procedures

```
WaitForKey(ch)                      // wait for a key pressed

// Procedures
InputVar(V,prompt,default)          // Display "prompt : " and ask for a value
                                    // if A is a table, use InputTable instead
InputTable(V,prompt,default)        // Input multiple values, in row x col order
InputTableRC(V,row,col,prompt,default)  // Input value at row,col
InputTableXY(V,x,y,prompt,default)  // Input value at x,y

// functions
InputS(prompt,default)              // Input a string, return a string, automatically converted to
                                    // number or boolean depends on the receiving variable type

// EditTable(V,prompt,default)      // display a form to edit table values
```

3. Math Functions

(n = integer, x = floating number)

Integer Functions:

```
Round(x)            // Round to nearest integer
Trunc(x)            // Truncated to nearest lower integer
Int(x)              // Truncated to integer part
Ceil(x)             // Get nearest higher integer
Floor(x)            // Get nearest lower integer
Even(n)             // 0 if not even number, 1 if even number
Odd(n)              // 0 if not odd  number, 1 if odd  number
Fact(n)             // factorial : 1*2*3*n-1*n
Inc(n)              // n+1
Dec(n)              // n-1
RandomRange(n1)     // 0..n1-1
RandomRange(n1,n2)  // n1..n2
```

Float Functions :

```
Random()            // 0..1
Random(n1)          // 0..n1-1
Random(n1,n2)       // n1..n2
Abs(x)              // absolute value of x
```

```
Sign(x)          // sign of x : +1 if > 0, 0 if zero, -1 if < 0
Deg(x)           // Degree of a radiant angle
Rad(x)           // Radiant of a degree angle
Sqrt(x)          // square root of x
Sqr(x)           // x^2
Power(x,y)       // x^y
Exp(x)           // e^x
Log(x)           // Ln(x)/Ln(10)
Log(a,b)         // Ln(a)/Ln(b)
LogN(a,b)        // Ln(a)/Ln(b)
Log2(x)          // Log2(x)
Log10(x)         // Log10(x)
Ln(x)            // Ln(x)
Max(x1,x2,...)   // max of (x1,x2,...)
Min(x1,x2,...)   // min of (x1,x2,...)
Length(x1,x2,...) // Sqrt(x1^2 + x2^2 + ....)
Avg(x1,x2,...)   // (x1+x2+...)/n
Sum(x1,x2,...)   // x1 + x2 + ...
SumSqr(x1,x2,...) // x1^2 + x2^2 + ....
SRSS(x1.x2,...)  // Sqrt(x1^2 + x2^2 + ....)
CQC(x1,x2,...)   // CQC(x1,x2,...)
```

Linear Interpolation:

```
LinearInterp(x,x1,y1,x2,y2);  // Linear interpolation of y using x value
```

Trigonometric functions:

```
Sin(x)           // Sine of x, radian
Cos(x)           // Cosine of x, radian
Tan(x)           // Tangent of x, radian
Sec(x)           // Secant of x, radian
Cosec(x)         // Cosecant of x, radian
CoTan(x)         // CoTangent of x, radian
SinD(x)          // Sine of x, degree
CosD(x)          // Cosine of x, degree
TanD(x)          // Tangent of x, degree
SecD(x)          // Secant of x, degree
CosecD(x)        // Cosecant of x, degree
CoTanD(x)        // CoTangent of x, degree
Asin(x)          // ArcSine of x, radian
Acos(x)          // ArcCosine of x, radian
Atan(x)          // ArcTangent of x, radian
Asec(x)          // ArcSecant of x, radian
Acosec(x)        // ArcCoSecant of x, radian
Acotan(x)        // ArcCoTangent of x, radian
SinH(x)          // Sine Hyperbolic of x, radian
CosH(x)          // Cosine Hyperbolic of x, radian
TanH(x)          // Tangent Hyperbolic of x, radian
SecH(x)          // Secant Hyperbolic of x, radian
CosecH(x)        // Cosecant Hyperbolic of x, radian
CoTanH(x)        // Cotangent Hyperbolic of x, radian
AsinH(x)         // Arc Sine Hyperbolic of x, radian
AcosH(x)         // Arc Cos Hyperbolic of x, radian
AtanH(x)         // Arc Tangent Hyperbolic of x, radian
```

```
AsecH(x)          // Arc Secant Hyperbolic of x, radian
AcosecH(x)        // Arc Cosecant Hyperbolic of x, radian
AcoTanH(x)        // Arc CoTangent Hyperbolic of x, radian
```

Calculus :

Vector :

```
VecLen(a,b,c,...)    // Length    : sqrt(a^2 + b^2 + c^2 + ….)
VecSum(a,b,c,...)    // Sum       : a + b + c + ...
VecAvg               // Average   : (a + b + c + ….)/n
VecSumSqr            // SumSqr    : a^2 + b^2 + c^2 + ….
VecSRSS              // SRSS      : sqrt(a^2 + b^2 + c^2 + ….)
```

Matrix :

```
MATRIX_NULL(A)          // Matrix, all 0
MATRIX_UNIT(A)          // Matrix, all 1
MATRIX_RANDOM(A)        // Matrix, all random numbers
MATRIX_COL(A,n)         // Column n of matrix A
MATRIX_ROW(A)           // Row n of matrix A
MATRIX_TRIDIAG(A)       // Trdiagonal Matrix A
MATRIX_TRANSPOSE(A)     // Transpose of A
MATRIX_DETERMINANT(A)   // Determinant of A
MATRIX_INVERSE(A)       // Inverse of A
MATRIX_FACTORIZE(A)     // Factorize of A = L.D.L^T
MATRIX_SOLVER(A,X,Y)    // Solve A.x = y
```

Linear equation :

Eigen Value :

5. String Functions

```
Trim(s)           // remove pre and post spaces
UpperCase(s)      // change all characters to uppercase
LowerCase(s)      // change all characters to lowercase
Copy(s,p,n)       // copy a substring from location p, n characters
IntToStr(n)       // convert integer n to string
NumToStr(x)       // convert numeric x to string
TimeToStr(t)      // convert time t to string
DateToStr(d)      // convert date d to string
Length(s)         // length of a string
Pos(s1,s)         // find a string s1 inside s
Val(s)            // value of a string
StrToInt(s)       // string to integer
StrToNum(s)       // string to numeric
StrToTime(s)      // string to time
StrToDate(s)      // string to date
```

6. Graphic Functions

```
SCRWID            // Screen Width in pixels
SCRHGT            // Screen Height in pixels
```

```
SelectImage(objname)                          // Select Image from an object
PenMode(pm)                                   // Set Pen Mode
Color(color)                                  // Set color(color)
Sound(filename,duration,pitch)                // Play a wav file

Text(x,y,agl,fsize,fcolor,txt)                // Draw a string with angle agl, size fsize
Line(x1,y1,x2,y2,thick,style,color)           // Draw a line from x1,y1 to x2,y2
Rect(x1,y1,x2,y2,thick,style,color,fillstyle,fillcolor)   // Draw a rectangle from x1,y1 to x2,y2
Circle(x,y,r[,thick,style,color,fillstyle,fillcolor)      // Draw a circle using x,y,r
Ellipse(x,y,rx,ry,thick,style,color,fillstyle,fillcolor)  // Draw an ellipse using x,y,rx,ry
Polygon(x,y,r,n,startagl,thick,style,color,fillstyle,fillcolor)  // Draw a n-sides polygon
```

Pen Mode	**Pixel color**
pmCopy	Pen color specified in the Color property
pmXor	Combination of colors in either pen or canvas background, but not both
pmNotCopy	Inverse of pen color
pmNotXor	Inverse of pmXor: combination of colors in either pen or canvas background, but not both
pmBlack	Always black
pmWhite	Always white
pmNop	Unchanged
pmNot	Inverse of canvas background color
pmMerge	Combination of pen color and canvas background color
pmNotMerge	Inverse of pmMerge: combination of pen color and canvas background color
pmMask	Combination of colors common to both pen and canvas background
pmNotMask	Inverse of pmMask: combination of colors common to both pen and canvas bkg
pmMergePenNot	Combination of pen color and inverse of canvas background
pmMaskPenNot	Combination of colors common to both pen and inverse of canvas background
pmMergeNotPen	Combination of canvas background color and inverse of pen color
pmMaskNotPen	Combination of colors common to both canvas background and inverse of pen

7. File Functions

```
OpenInpText(fname)                            // Open input text file
CloseInpText(fname)                           // Close input text file
Readln(v1,v2,v3,...)                          // Read a line from input text file

OpenOutText(fname)                            // Open output text file
CloseOutText(fname)                           // Close output text file
Writeln(v1,v2,v3,...)                         // Write several variables
```

8. Multimedia and GUI

```
PlaySound(fname)                              // Play a wav file
PlayVideo(fname)                              // Play a video file
Waitforkey(char)                              // Wait for a key pressed
Display(s)                                    // Display a string and wait for user to press OK
Print(s)                                      // Print a string or variable to output screen
Print A;                                      // Print a string or variable to output screen
```

9. Data Acquisition and Sensors

10. Structural Analysis Module

Structural Analysis Module in BALI is using SANS FEM Module.
A structure model including node coordinates, supports, members, and joint/member loads can be defined using simple syntax.
Several element types area available: Truss, Frame, QUAD4
Several analysis methods are provided : Static Analysis, Dynamic Spectrum Response Analysis, and Linear Time History Analysis.
After analysis, deformed shape, mode shape, node displacements, support reactions, and member forces can be displayed using table and graphics.

SANSFEM COMMANDS:

SANS_INIT(NLS,NLC,R1,R2,R3,R4,R5,R6, Flag1)

Initialize SANSFEM :
NLS = number of load combination
NLC = number of load cases
R1..R6 = restraint for direction 1..6 (0=free, 1=restrained)
Flag1 = debugging flag

SANS_PROP(matname,sctname : string; Es,G,PR,C,Ywg,Ag,Av,Aw,Ix,Iy,Iz : double;
scttype : integer; d,bw,bf,tf,bb,tb,a : double;
mopt,comp,nb : integer; n,bc,tphc,dc,fc,fy : double)

Set material properties :

matname	: material name
sctname	: section name
Es	: elastic modulus
G	: shear modulus
PR	: poisson ratio
C	: thermal coefficient
Ywg	: unit weight
Ag	: gross area
Av	: shear area, major direction
Aw	: weight area
Ix	: inertia in x direction
Iy	: inertia in y direction
Iz	: inertia in z direction
scttype	: section type
d	: section depth
b	: section web width = tw
bf	: section top flange width
tf	: section flange thickness
bb	: section bottom flange width
tb	: section bottom flange thickness
a	: section lip
mopt	: material option
comp	: composite option
nb	: number of rebar
n	: n= Es/Ec
bc	: encased width
tphc	: encased height
dc	: distance between centroid of slab to centroid of beam
fc	: concrete strength
fy	: steel yield strength

SANS_NODE(id, x,y,z)

Define Node coordinates:

id	: node id
x	: x coordinate
y	: y coordinate
z	: z coordinate

SANS_SUPPORT(id,j,R1,R2,R3,R4,R5,R6)

Define Support Condition:

id	: support id
j	: node id
R1	: restraint in global 1 direction
R2	: restraint in global 2 direction
R3	: restraint in global 3 direction
R4	: restraint in global 4 direction
R5	: restraint in global 5 direction
R6	: restraint in global 6 direction

SANS_TRUSS(name,n1,n2,mat)

Define a truss element data

name	: truss element name, can be a number or alphabet + number
n1	: first node n1
n2	: second node n2
mat	: material index

SANS_TRUSSXYZ(name,x1,y1,z1,x2,y2,z2,mat)

Define a truss element data

name	: truss element name, can be a number or alphabet + number
n1	: first node n1
n2	: second node n2
mat	: material index

SANS_FRAME(name, n1,n2, alpha,rgzf1,rgzf2,mat,rel,rge)

Define a frame Element Data

name	: truss element name, can be a number or alphabet + number
n1	: first node n1
n2	: second node n2
mat	: material index
alpha	: section rotation angle
rgzf1	: rigid end offset factor 1
rgzf2	: rigid end offset factor2
rel	: release option
rge	: rigid end offset option

SANS_FRAMEXYZ(name, x1,y1,z1,x2,y2,z2, alpha,rgzf1,rgzf2,mat,rel,rge)

Define a frame Element Data

name	: truss element name, can be a number or alphabet + number
x1,y1,z1	: first node coordinates
x2,y2,z2	: second node coordinates
mat	: material index
alpha	: section rotation angle
rgzf1	: rigid end offset factor 1
rgzf2	: rigid end offset factor2

rel : release option
rge : rigid end offset option

SANS_QUAD4 or SANS_SHELL(name,n1,n2,n3,n4,tp,mat)

Define a shell quad4 4-node Element Data
name : shell element name, can be a number or alphabet + number
n1..n4 : node index
tp : shell thickness
mat : material index

SANS_QUAD4XYZ or SANS_SHELLXYZ(name,x1,y1,z1,...x4,y4,z4,tp,mat)

Define a shell quad4 4-node Element Data
name : shell element name, can be a number or alphabet + number
xi,yi,zi : node i coordinate
tp : shell thickness
mat : material index

SANS_SOLID(name,n1,n2,n3,n4,n5,n6,n7,n8,tp,mat)

Define a brick solid 8-node Element Data
name : shell element name, can be a number or alphabet + number
n1..n8 : node index
tp : shell thickness
mat : material index

SANS_SOLIDXYZ(name,x1,y1,z1,...x8,y8,z8,tp,mat)

Define a brick solid 8-node Element Data
name : shell element name, can be a number or alphabet + number
xi,yi,zi : node i coordinate
tp : shell thickness
mat : material index

SANS_LOADCOMB(ldcomb,name,SW,DL,LL,EQX,EQZ,WX,WZ,EPX,EPZ,PS)

Add a Load combination
ldcomb : load combination id
name : load combination name
SW : load factor for Self Weight
DL : load factor for Dead Load
LL : load factor for Live Load
EQX : load factor for Earthquake in X direction
EQZ : load factor for Earthquake in Z direction
WX : load factor for Wind in X direction
WZ : load factor for Wind in Z direction
EPX : load factor for Earth pressure in X direction
EPZ : load factor for Earth pressure in Z direction
PS : load factor for prestressing load

SANS_JLOAD(name,ldcase,j,fx,fy,fz,mx,my,mz)

Add a Joint Load Data
name : joint load name
ldcase : joint load case id
j : node id
fx : force joint load in global X direction
fy : force joint load in global Y direction
fz : force joint load in global Z direction

mx : moment load in global X direction
my : moment load in global Y direction
mz : moment load in global Z direction

SANS_JLOADXYZ(name,ldcase,x,y,z,fx,fy,fz,mx,my,mz)

Add a Joint Load Data

name : joint load name
ldcase : joint load case id
x,y,z : coordinate of joint load
fx : force joint load in global X direction
fy : force joint load in global Y direction
fz : force joint load in global Z direction
mx : moment load in global X direction
my : moment load in global Y direction
mz : moment load in global Z direction

SANS_TLOAD(name,lc,m,typ,np,p1,p2,p3,p4,p5)

Add a truss member Load Data

name : truss member load name
lc : load case id
m : member id
typ : member load type
np : number of load parameters
p1..p5 : member load parameters

SANS_FLOAD(name,lc,m,typ,np,p1,p2,p3,p4,p5)

Add a frame member Load Data

name : frame member load name
lc : load case id
m : member id
typ : member load type
np : number of load parameters
p1..p5 : member load parameters

SANS_DOF

Compute active DOF index

SANS_SHOWDOF

Show generated active DOF

SANS_SKYLINE

Compute stiffness matrix variable band

SANS_SHOWFROW

Show FROW values = first nonzero row of columns of global stiffness matrix

SANS_SHOWDIAG

Show diagonal entries of global stiffness matrix

SANS_STIFF

Compute global stiffness matrix

SANS_SHOWSTIFF

Show global stiffness matrix

SANS_LOAD(ldc)
Compute global load vector for loadcase ldc

SANS_MASS
Compute global mass vector

SANS_FACTORIZE
Factorize the global stiffness matrix to L.D.L^ format

SANS_SHOWXVEC
Show global displacement X vector

SANS_SOLVER(ldc)
Solve global displacement vector x for loadcase ldc F : K.x = F

SANS_EIGEN
Compute eigen problem

SANS_DYNRES
Compute Dynamic Response Spectrum Analysis

SANS_STOREDISP(ldcomb)
Store displacement vector for loadcomb ldcomb

SANS_COMPUTEFORCE(ldcomb)
Compute element forces for load combination ldcomb

SANS_STOREREACT(ldcomb)
Compute reactions for load combination ldcomb

SANS_GETDISP(n,op,lsc)
Get a node displacement
n : node id
op : option, 0=ldcase, 1=ldcomb
lsc : ldcase or ldcomb

SANS_GETREACT(n,op,lsc)
Get a nodal reaction
n : node id
op : option, 0=ldcase, 1=ldcomb
lsc : ldcase or ldcomb

SANS_GETTRUSSF(m,op,lsc)
Get a truss member local forces
m : member id
op : option, 0=ldcase, 1=ldcomb
lsc : ldcase or ldcomb

SANS_GETFRAMEF(m,op,lsc)
Get a frame member local forces
m : member id
op : option, 0=ldcase, 1=ldcomb
lsc : ldcase or ldcomb

SANS_GETQUAD4F(m,op,lsc)

Get a quad4 member local forces
m : member id
op : option, 0=ldcase, 1=ldcomb
lsc : ldcase or ldcomb

SANS_REPORT

Generate Analysis Report

SANS_GRAPH(ldcomb,ldcase)

Show graphics
ldcomb : load comb index, if zero, use individual ldcase
ldcase : locad case index

MIDI Instrument Code Number

Piano:
1 Acoustic Grand Piano
2 Bright Acoustic Piano
3 Electric Grand Piano
4 Honky-tonk Piano
5 Electric Piano 1
6 Electric Piano 2
7 Harpsichord
8 Clavinet

Chromatic Percussion:
9 Celesta
10 Glockenspiel
11 Music Box
12 Vibraphone
13 Marimba
14 Xylophone
15 Tubular Bells
16 Dulcimer

Organ:
17 Drawbar Organ
18 Percussive Organ
19 Rock Organ
20 Church Organ
21 Reed Organ
22 Accordion
23 Harmonica
24 Tango Accordion

Guitar:
25 Acoustic Guitar (nylon)
26 Acoustic Guitar (steel)
27 Electric Guitar (jazz)
28 Electric Guitar (clean)
29 Electric Guitar (muted)
30 Overdriven Guitar
31 Distortion Guitar
32 Guitar harmonics

Bass:
33 Acoustic Bass
34 Electric Bass (finger)
35 Electric Bass (pick)
36 Fretless Bass
37 Slap Bass 1
38 Slap Bass 2
39 Synth Bass 1
40 Synth Bass 2

Strings:
41 Violin
42 Viola
43 Cello
44 Contrabass
45 Tremolo Strings
46 Pizzicato Strings
47 Orchestral Harp
48 Timpani

Strings (continued):
49 String Ensemble 1
50 String Ensemble 2
51 Synth Strings 1
52 Synth Strings 2
53 Choir Aahs
54 Voice Oohs
55 Synth Voice
56 Orchestra Hit

Brass:
57 Trumpet
58 Trombone
59 Tuba
60 Muted Trumpet
61 French Horn
62 Brass Section
63 Synth Brass 1
64 Synth Brass 2

Reed:
65 Soprano Sax
66 Alto Sax
67 Tenor Sax
68 Baritone Sax
69 Oboe
70 English Horn
71 Bassoon
72 Clarinet

Pipe:
73 Piccolo
74 Flute
75 Recorder
76 Pan Flute
77 Blown Bottle
78 Shakuhachi
79 Whistle
80 Ocarina

Synth Lead:
81 Lead 1 (square)
82 Lead 2 (sawtooth)
83 Lead 3 (calliope)
84 Lead 4 (chiff)
85 Lead 5 (charang)
86 Lead 6 (voice)
87 Lead 7 (fifths)
88 Lead 8 (bass + lead)

Synth Pad:
89 Pad 1 (new age)
90 Pad 2 (warm)
91 Pad 3 (polysynth)
92 Pad 4 (choir)
93 Pad 5 (bowed)
94 Pad 6 (metallic)
95 Pad 7 (halo)
96 Pad 8 (sweep)

Synth Effects:
97 FX 1 (rain)
98 FX 2 (soundtrack)
99 FX 3 (crystal)
100 FX 4 (atmosphere)
101 FX 5 (brightness)
102 FX 6 (goblins)
103 FX 7 (echoes)
104 FX 8 (sci-fi)

Ethnic:
105 Sitar
106 Banjo
107 Shamisen
108 Koto
109 Kalimba
110 Bag pipe
111 Fiddle
112 Shanai

Percussive:
113 Tinkle Bell
114 Agogo
115 Steel Drums
116 Woodblock
117 Taiko Drum
118 Melodic Tom
119 Synth Drum

Sound effects:
120 Reverse Cymbal
121 Guitar Fret Noise
122 Breath Noise
123 Seashore
124 Bird Tweet
125 Telephone Ring
126 Helicopter
127 Applause
128 Gunshot

Table of Musical Frequencies

Note	Frequency
C	130.82
C#	138.59
D	146.83
D#	155.56
E	164.81
F	174.61
F#	185
G	196
G#	207.65
A	220
A#	233.08
B	246.94

Note	Frequency
C	261.63
C#	277.18
D	293.66
D#	311.13
E	329.63
F	349.23
F#	369.99
G	392
G#	415.3
A	440
A#	466.16
B	493.88

Note	Frequency
C	523.25
C#	554.37
D	587.33
D#	622.25
E	659.26
F	698.46
F#	739.99
G	783.99
G#	830.61
A	880
A#	932.33
B	987.77

Note	Frequency
C	1046.5
C#	1108.73
D	1174.66
D#	1244.51
E	1318.51
F	1396.91
F#	1479.98
G	1567.98
G#	1661.22
A	1760
A#	1864.66
B	1975.53
C	2093.00