

EE300 Summer Practice Report

Student Name : Damla ÖZGÜL
Student ID # : 138993-1
SP Date : 12/06/2006 to 07/07/2006
Submission Date : 29/09/2006

SP Company Name : ASELSAN
Company Division : Microwave and System Technologies
Division, Test Engineering
Company Location : Macunköy, Ankara
Related Field : Control

Middle East Technical University
Department of Electrical and Electronics Engineering

TABLE OF CONTENT

1. INTRODUCTION

2. DESCRIPTION OF THE COMPANY

2.1 Company Name

2.2 Company Location

2.3 General Description of the Company

- Organizational Structure
- Mission & Vision
- General Objectives
- Number and Duties of Engineers Employed

2.4 Shareholders of the Company

2.5 Main Area of Business

2.6 Brief History of Company

3. RADAR CHARACTERISTICS:

3.1 WHAT IS A RADAR?

3.2 BASIC RADAR TERMS

3.3 WHAT IS RADSIM?

3.4 MAIN PURPOSE OF THE WORK

4. GRAPHING THE RESULTS OF RADSIM WITH VEE

4.1 THE MAIN FUNCTION

4.2 DESIGNING PROCEDURE

4.2.a. 'data_seperate' function

4.2.b. 'b' function

4.2.c. 'statistic' function

4.2.d. 'setting' function

4.2.e. 'sizes_n' function

4.2.f. 'min_size' function

4.2.g. 'graph' function

4.2.h. 'first_setting' function

4.3 IMPROVEMENT OF THE PROGRAM

4.3.a. 'Buttons' function

5. GRAPHING THE RESULTS OF RADSIM WITH VISUAL C++

5.1 OPEN FILE BUTTON

5.2 START BUTTON

5.2.a. Reading the file

5.2.b. Seperating the radars' data

5.2.c. Graphing the radars

5.3 EXIT BUTTON

6. CONCLUSION

7. REFERENCES

1. INTRODUCTION:

I have performed my second year Summer Practice in ASELSAN which is the leading multi-product electronics company of Turkey. ASELSAN designs, develops and manufactures modern electronic systems not only for military but also for professional customers. My practice lasted totally 4 weeks, started at 12.06.2006 and ended in 07.07.2006. The division where I have performed my work is the Test Engineering Department (TMM) which is the part of Microwave and System Technologies Division (MST). (I will explain the divisions of ASELSAN in the description of the company part in vivid detail.) The Test Engineering Department is also divided into two groups; Radar Electronic War and Programming. I have performed my SP in Radar Electronic War (TMM-REH) group.

In TMM-REH group, the engineers are working on the advanced computer controlled automatic testing systems and some simulation software systems. I have worked with one of this simulation softwares (RADSIM) in which the user can simulate radars. Briefly what I have done during the SP was to show graphically the results of this simulation software, obtained from a text file, using the programs Agilent VEE and Visual C++. I can entitle my work as 'Projection of Radar Pulses'. For this purpose I have got a general information about radars and the ASELSAN radar simulation software. Also I have learned the programming languages while I am doing the project. And finally I have made a presentation of my project to the TMM-REH group.

In this report, a detailed description of the things that I have done and observed during the summer practice is included. It begins with the description of the company in which the organizational structure of the company, main areas of the business and a brief history of the company are explained. Then the main part of the report which includes the works related with the 'Projection of Radar Pulses' follows the description part of the report. And finally, a report is concluded with a conclusion part. Also an appendix is included as a reference text.

2. DESCRIPTION OF THE COMPANY

2.1. COMPANY NAME

ASELSAN Electronic Industries Inc.

2.2. COMPANY LOCATION

In Ankara ASELSAN has two facilities, which are Macunköy and Akyurt Facilities. At Macunköy Facilities, Communications (HC) Division and Microwave and System Technologies (MST) Division are located whereas at Akyurt Facilities, Microelectronics, Guidance and Electro-Optics (MGEO) Division is located.

Macunköy Facilities

Address: Mehmet Akif Ersoy Mah.16. Cadde No: 16, Macunköy, 06370, Ankara, Türkiye

Phone: (90) 312-592 10 00

Fax: (90) 312-354 13 02 / (90) 312-354 26 69

Akyurt Facilities

Address: P.K. 30, Etlik, 06011, Ankara, Türkiye

Phone: (90) 312-847 53 00

Fax: (90) 312-847 53 20

2.3. GENERAL DESCRIPTION OF THE COMPANY

During the Cyprus War (1974), lots of soldier in Turkish army died because of the problems in the electronic communication systems. In fact a fighter aircraft from Turkish Army bombed a Turkish fighter ship. This event was a milepoint in the foundation of ASELSAN. At the end of 1975, the Turkish Armed Forces Foundation founded ASELSAN to produce tactical military radios and defense electronic systems for the Turkish Army. In early 1979 following an investment and infrastructure establishment period, ASELSAN started its production, at Macunköy-Ankara facilities. From the day it was founded up to now, ASELSAN has expanded its product and customer spectrum, with qualified personnel, high technology, and knowledge.

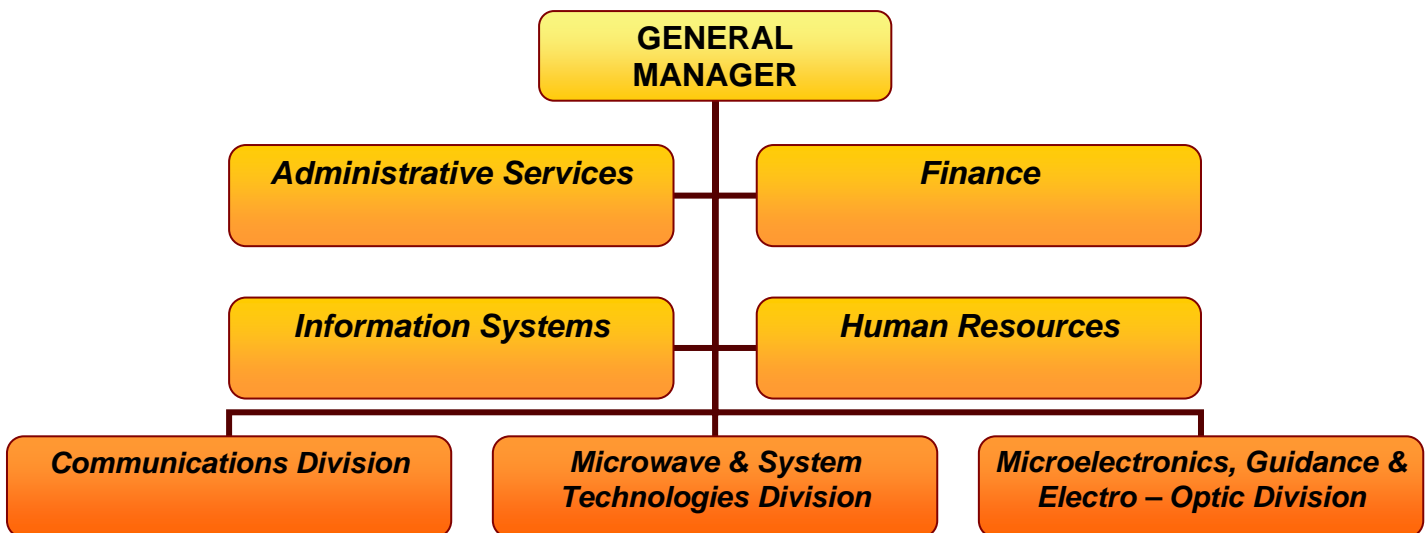
ASELSAN is the leading multi-product electronics company of Turkey which works on designing, developing and manufacturing modern electronic systems for military and professional customers. According to the field of activities ASELSAN has been organized in three main divisions:

- 1) Communications (HC) Division
- 2) Microwave and System Technologies (MST) Division
- 3) Microelectronics, Guidance and Electro-Optics (MGEO) Division

The Communications Division and Microwave and System Technologies Division have high-technology and automated infrastructure in engineering and production at Macunköy facilities. Electronic production includes surface mount technology, multilayer and flexible printed circuit boards, mechanical and mould productions, system integration and test fields. While Communications Division's main product spectrum covers **military and professional communications systems**, Microwave and System Technologies Division's main operations are focused on **radar, electronic warfare and command-control systems**, making these divisions evolving technology centers in their fields. Equipped with high technology engineering, automatic production and test equipment, Microelectronics, Guidance and Electro-Optics Division manufactures **hybrid microelectronic circuits, night vision equipment, thermal cameras, laser ranger/designators and inertial navigation systems** at Akyurt facilities.

In all divisions, methodologies complying with military standards and ISO-9001 are successfully applied using computer aided design (CAD), computer aided engineering (CAE) and computer aided manufacturing (CAM) technologies.

- **Organizational Structure:**



- **Mission & Vision:**

MISSION

ASELSAN's mission is, by following up advanced technology, to meet the electronic product and system needs of Turkey in the most favorable price-time-quality conditions to use this knowledge accumulation in export opportunities, and thus, to obtain continuity and development under all conditions.

VISION

ASELSAN's vision is to become the best in Turkey in its domain by improving the successful status gained within the country and abroad, and as a straightforward and a reliable company to obtain customer satisfaction at home and throughout the world.

- **General Objectives:**

- ❖ To create the environment in which the military and civil electronic product and system needs of our country will be met by technology production.
- ❖ To realize exportation in high technology products by achieving the level of competitive capability both in terms of quality and price in international markets.
- ❖ To balance the production between military and professional electronic products and thus obtain ASELSAN's continuity and development under every condition.
- ❖ To provide the necessary potential, to realize the production for the topics that ought to be national like electronic warfare, encryption and critical software.
- ❖ To be a technology center that can produce and design military/ professional electronic products and systems in the determined operating fields.
- ❖ To give primary importance to R&D studies.
- ❖ To maintain technology based, planned and healthy growth.
- ❖ To obtain maximum customer satisfaction for each product and to comply to international quality standards.
- ❖ To provide a peaceful and secure working environment and in direction of the company objectives of ASELSAN for the personnel.
- ❖ To devote social values and commercial ethics, and to preserve the customer rights and the natural environment while realizing the company objectives.

- **Number and Duties of Engineers Employed:**

According to company regulations, we are not allowed to give detailed information about this topic. However, by December 2005, it is reported that there are 2997 employees, 1230 of which are engineers, working in ASELSAN.

2.4. SHAREHOLDERS OF THE COMPANY

- **Shareholders:**

Turkish Armed Forces Foundation	84.58 %
<i>AXA OYAK Insurance</i>	<i>0.12 %</i>
<i>Other Shareholders</i>	<i>15.30 %</i>

- **Participations:**

<i>ASELSAN'S SHARE</i>	<i>SHARE RATE (%)</i>
<i>ROKETSAN</i>	<i>15</i>
<i>ASPIŁSAN</i>	<i>1</i>
<i>TÜLOMSAŞ</i>	<i>0.476</i>
<i>ASELSAN-BAKÜ</i>	<i>100</i>
<i>HAVAŞ</i>	<i>0.051</i>
<i>MİKES</i>	<i>72</i>
<i>ASNET</i>	<i>95</i>

2.5. MAIN AREA OF BUSINESS

ASELSAN is a high technology multiproduct electronics corporation that designs, develops, produces and services products and systems for military and professional applications.

In ASELSAN the latest electronic, electro-optics and mechanical technologies have been carried out in product development by taking advantage of the computer aided development and production infrastructure.

- ***Communications Division***
- ***Microwave and System Technologies Division***
- ***Microelectronics, Guidance and Electro-Optics Division***

Since I have performed my SP in Microwave and System Technologies Division, I want to give brief information about the division and organization of this division.

Briefly in Microwave and System Technologies Division engineers are focused on radar, electronic warfare and command-control systems which are generally relating with the term “war”. Therefore the main customer of this division is military services. In this division engineers deal with designing new technologies for war systems and improving the technologies that are already used in Turkish Army.

The Microwave and System Technologies Division continues its production and delivery activities under the following fields:

- Defence and Weapon Systems
- Electronic Warfare and Intelligence
- Command Control Systems
- Radar Systems
- Control and Automation Systems

In Defence and Weapon Systems, superior capabilities to its customers with its air defence systems, fire control systems for main battle tanks, fire control systems for self-propelled weapons are produced. On the other hand, Electronic Warfare and Intelligence part tailors specific integrated electronic warfare and intelligence systems for military and professional users. In Command and Control & Control and Automation Systems, engineers develop Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) systems and equipment to meet the requirements of the tactical battlefield in the 21st Century. Finally in Radar Systems, ASELSAN manufactures its own design radars and radar Technologies.

2.7. BRIEF HISTORY OF THE COMPANY

- ASELSAN was established in **1975**, to meet the communications electronics requirements of the Turkish Armed Forces.
- Mr. Dr. M. Hâcim KAMOY has appointed as general manager on January **1976**.
- ASELSAN completed its primary investment at Macunköy facilities in **1979** and within a year, the first production activities were initiated.
- In **1980** military man pack and tank radio production had started and the first delivery had been realized.
- ASELSAN designed its first hand-held radios and bank alarm systems in **1981**.
- **1983** was the year for the first export business. By the end of **1983**, ASELSAN had 1,434 personnel including 186 engineers.
- ASELSAN enlarged its product spectrum between **1982-1985**. TBX exchange systems, field telephones, computer controlled central systems and laser range finders were among the new products.
- ASELSAN contributed to the defense power of the Turkish Armed Forces by its Electronic Warfare and Data Terminals in **1986**.
- In **1987** ASELSAN participated in NATO Joint Production Project of Stinger missiles and started the necessary investment for the production of thick film hybrid circuits.
- In **1988** ASELSAN produced the first avionic equipment: Inertial Navigation Systems for F-16 aircrafts. ASELSAN quality system was certified in accordance with AQAP-4 standards. Electronic Proximity Fuze contract signed with MOD in this year.
- In **1989** ASELSAN realized the first technology transfer programme to Pakistan. Combat area tactical radios production started in NRTC facilities in Pakistan under ASELSAN's licence.
- In **1990** ASELSAN became 47th of the European Defense Electronics Companies. Field Artillery Battery Fire Control System and TV Transmitter production started. ASELSAN had 2,000 personnel including 330 engineers.
- In **1991** ASELSAN was organized under 3 divisions as Communications Division, Microwave and System Technologies Division and Microelectronics, Guidance and Electro-Optics Division regarding the projects within the field activity. ASELSAN was chosen as the 127th of the world defense companies by International Defense Magazine. ASELSAN, by its treatment facilities, won the prize of the best company which cares for its environment.

- The most important characteristic of **1992** was the addition of the Radar Systems to ASELSAN's product spectrum. TQM implementations have accelerated since 1992.
- In **1993** ASELSAN made a big achievement by establishing the Electro-Optics Technology Center at Akyurt Facilities. ASELSAN quality system was certified in accordance with ISO 9001 standards.
- In **1994** ASELSAN quality system was revised as AQAP-1 standards. Have Quick Radio production started.
- In **1995** ASELSAN engineers completed design activities of ASELSAN's first consumer product, the Mobile Phone. ASELSAN increased its exports to 19 countries. The Railway Transport System project in Power Electronics area started.
- In **1996** ASELSAN quality system was revised as AQAP-110 standards. TASMUS contract which will provide a communication system with recent technology to Turkish Armed Forces was signed.
- In **1997** with its Mobile Phone-1919 which is completely designed by ASELSAN's engineers; Turkey has taken its position among the first nine countries who designed its own mobile phone.
- In **1998**, ASELSAN manufactured and delivered various new equipment. Thermal camera systems, thermal weapon sights, thermal imaging equipment and laser designators were produced for the needs of the Turkish Armed Forces. Design of Automatic Toll Collection System was completed and production was started. A contract for National Monitoring System was signed with the General Directorate of Radio Communications. In addition, ASELSAN has received the "Approved Producer" certificate from the American Government for the production of LN-93 Inertial Navigation Systems.
- **1999** was a year in which various equipment designed by ASELSAN has gained considerable success. During the live fire tests of Pedestal Mounted Stinger System, 100% success was achieved. On the other hand, design of new mobile phone was completed and Europe Approval was taken. In addition, ASELSAN has received new projects from Turkish MOD. Contracts have signed for the production of Air Defence Early Warning Command Control Systems, for the design and production of Electronic Warfare Systems and X-Band Satellite Communication Systems.
- Mr. Dr. M. Hâcim KAMOY who was the General Manager for 25 years, retired as for November **2000** and Mr. Necip Kemal BERKMAN has appointed as General Manager.

3. RADAR CHARACTERISTICS:

3.1. WHAT IS A RADAR?

The term 'radar' can be defined as an instrument used to detect precipitation by measuring the strength of the electromagnetic signal reflected back. Actually this term is a abbreviation of RAdio Detection And Ranging. The function of radar is to locate things by determining their range and angular position relative to the radar location and orientation. A radar determines the distance to some object (which we will call the target) by measuring the time it takes a propagated signal to travel to and from the target at the speed of light.

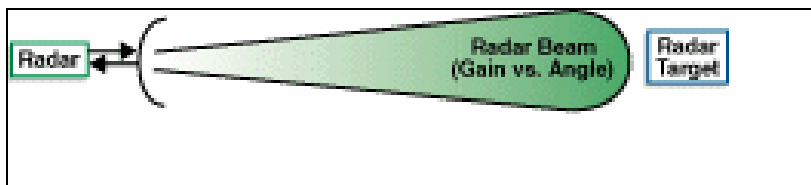


Figure 1

3.2. BASIC RADAR TERMS:

Pulse Width (PW): the duration of a single pulse of a radar

Pulse Repetition Interval (PRI) : the duration between two following consecutive pulses of a single radar

Time of Arrival (TOA) : the time when a pulse arrives

Effective Radiated Power (ERP) : simply the amplitude of a pulse

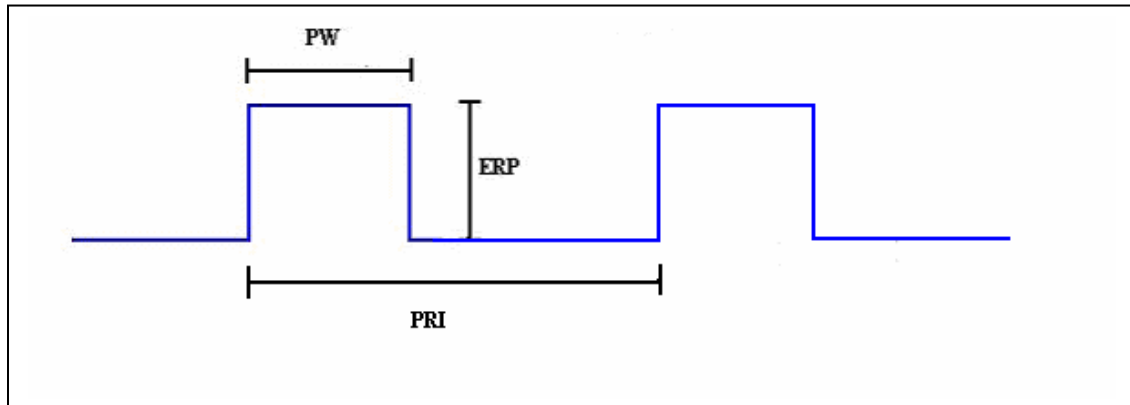


Figure 2

3.3. WHAT IS RADSIM? :

According to company regulations, I am not allowed to give detailed information about this topic. However, if I briefly explain RADSIM is a radar simulation program created by Aselsan TMM-REH engineers. It is a easy to use program, although it has required a detailed study in the designing procedure. In the program users can create different radars with different pulse intervals, different starting times etc... Users only fill the blanks which are related with the radars' pulses, then the program shows all datas of the radars as a pwd file as shown in the *Figure 3*.

In the 'pwd' file shown in *Figure 3*, it can be understood from the radar no.s that three radars exist; radar1, radar2 and radar3. All of them have a pulse width of 10usec. The first radar's first pulse arrives at $t=0$, ends at $t=10$ while the second pulse arrives at $t=20$. Therefore the pulse interval, which can be defined as the time between the ending time of a pulse and the beginning time of the next pulse, is 10usec ($=20-10$). This is only an example of analysing the file. All the pulses of three radars can be specified like that. Also from the figure it can be understood that the amplitude of the first radar is -10dBm, the amplitude of the second radar is -30dBm and the amplitude of the third radar is -45dBm.

DD Radar No	Pulse No	PW (usec)	PI (usec)	TOA (usec)	AOA (deg)	ERP (dBm)	Modulation Mod.	Frequency (GHz)
1	1	10.000	10.000	0.000	0.000	-10.000	None	10.000
1	2	10.000	10.000	20.000	0.000	-10.000	None	10.000
1	3	10.000	10.000	40.000	0.000	-10.000	None	10.000
1	4	10.000	10.000	60.000	0.000	-10.000	None	10.000
1	5	10.000	10.000	80.000	0.000	-10.000	None	10.000
1	6	10.000	10.000	100.000	0.000	-10.000	None	10.000
1	7	10.000	10.000	120.000	0.000	-10.000	None	10.000
1	8	10.000	10.000	140.000	0.000	-10.000	None	10.000
1	9	10.000	10.000	160.000	0.000	-10.000	None	10.000
1	10	10.000	10.000	180.000	0.000	-10.000	None	10.000
1	11	10.000	1.000	200.000	0.000	-10.000	None	10.000
3	3	10.000	9.000	211.000	0.000	-45.000	None	10.000
1	12	10.000	20.000	230.000	0.000	-10.000	None	10.000
1	13	10.000	20.000	260.000	0.000	-10.000	None	10.000
1	14	10.000	5.000	290.000	0.000	-10.000	None	10.000
2	5	10.000	5.000	305.000	0.000	-30.000	None	10.000
1	15	10.000	20.000	320.000	0.000	-10.000	None	10.000
1	16	10.000	1.000	350.000	0.000	-10.000	None	10.000
3	6	10.000	9.000	361.000	0.000	-45.000	None	10.000
1	17	10.000	20.000	380.000	0.000	-10.000	None	10.000
1	18	10.000	20.000	410.000	0.000	-10.000	None	10.000
1	19	10.000	20.000	440.000	0.000	-10.000	None	10.000
1	20	10.000	20.000	470.000	0.000	-10.000	None	10.000
1	21	10.000	5.000	500.000	0.000	-10.000	None	10.000
2	8	10.000	15.000	515.000	0.000	-30.000	None	10.000
1	22	10.000	11.000	540.000	0.000	-10.000	None	10.000
3	10	10.000	9.000	561.000	0.000	-45.000	None	10.000
1	23	10.000	30.000	580.000	0.000	-10.000	None	10.000
1	24	10.000	30.000	620.000	0.000	-10.000	None	10.000
1	25	10.000	30.000	660.000	0.000	-10.000	None	10.000
1	26	10.000	1.000	700.000	0.000	-10.000	None	10.000
3	13	10.000	4.000	711.000	0.000	-45.000	None	10.000
2	11	10.000	5.000	725.000	0.000	-30.000	None	10.000
1	27	10.000	11.000	740.000	0.000	-10.000	None	10.000
3	14	10.000	9.000	761.000	0.000	-45.000	None	10.000
1	28	10.000	5.000	780.000	0.000	-10.000	None	10.000
2	12	10.000	15.000	795.000	0.000	-30.000	None	10.000
1	29	10.000	30.000	820.000	0.000	-10.000	None	10.000
1	30	10.000	30.000	860.000	0.000	-10.000	None	10.000
1	31	10.000	1.000	900.000	0.000	-10.000	None	10.000
3	17	10.000	14.000	911.000	0.000	-45.000	None	10.000
2	14	10.000	5.000	935.000	0.000	-30.000	None	10.000
1	32	10.000	1.000	950.000	0.000	-10.000	None	10.000
3	18	10.000	29.000	961.000	0.000	-45.000	None	10.000
1	33	10.000	1.000	1000.000	0.000	-10.000	None	10.000

Figure 3

3.4. MAIN PURPOSE OF THE WORK :

As mentioned in the 3.3 part RADSIM shows the radars' data in the text format, however it is hard to understand and visualize in mind. That is, if these radars' pulses could be represented graphically , that would be convenient and useful for users of the program. Actually that is the thing I was supposed to do. I have worked on representing the radars graphically using the necessary variables of the 'pwd' file which are radar number, pulse width, and time of arrival. In fact simply what I had done was to show the radars' pulses like shown in Figure 2 using the 'pwd' file in Figure 3. I should have completed this work by using two software programs; VEE and Visual C++. First I will explain the representation with VEE

4. GRAPHING THE RESULTS OF RADSIM WITH VEE :

At the beginning of my work I should have made a plan of what I have needed to do. First, the pwd file should have been read by the program, then since the radars' datas are disorganized they should have been separated, then for each radar the data should have been considered and they should have been transferred to the graph as dots. Finally for convenience some buttons can be added. These buttons would satisfy user to see the radars separately.

4.1. THE MAIN FUNCTION :

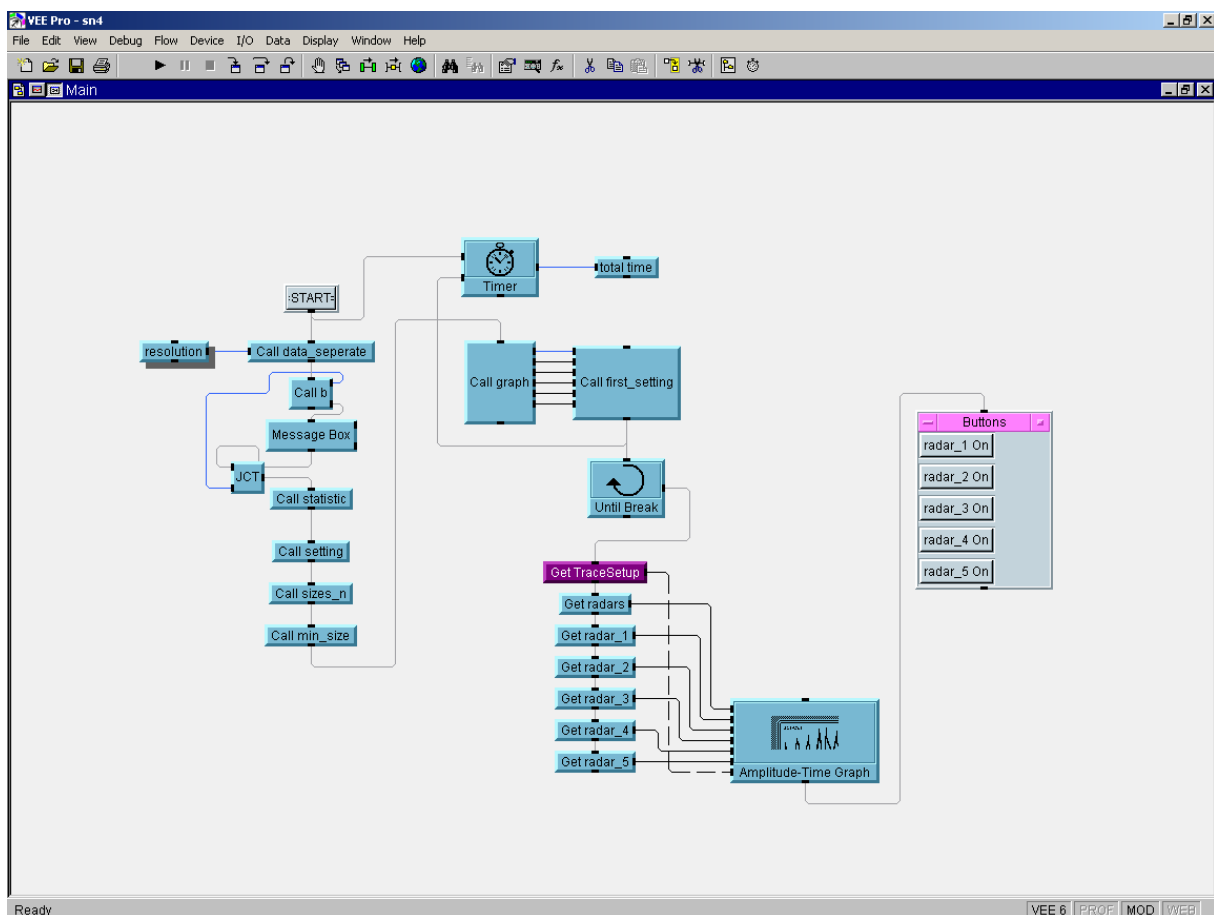


Figure 4 – Detailed View

In **Figure 4** the detailed view of the program is shown. This is the main body of the program and it contains all the functions used in the program. However this is not the view that the user of the program works on , the user only sees the panel view of the program. An example of the panel view is shown in **Figure 5**.

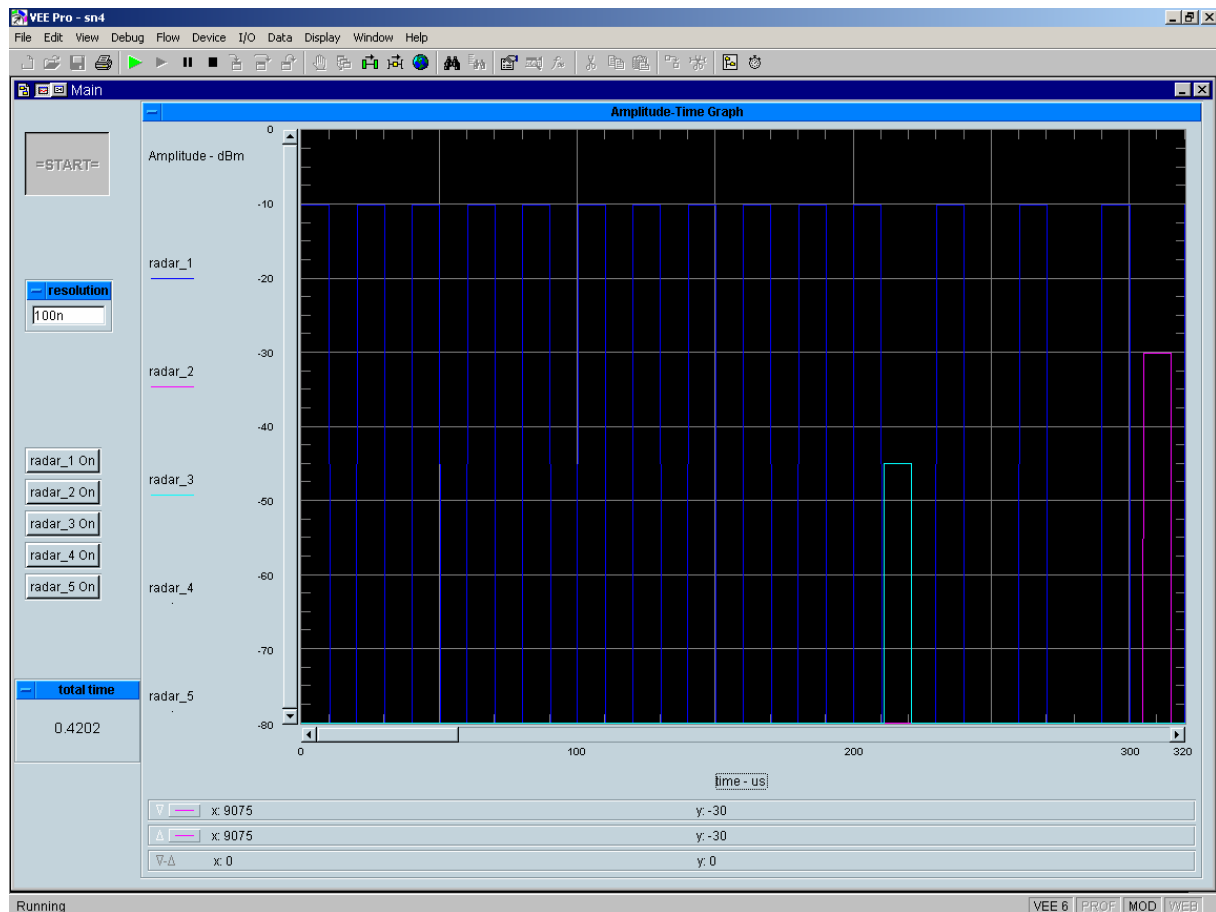


Figure 5 – Panel View

**Actually when I had completed designing this program, I worked on improving it for increasing the facilities of the program. Therefore in this report, I will first explain the designing procedure, then I will mention the improvement of the program with some buttons.

4.2. DESIGNING PROCEDURE :

• 4.2.a. 'data_seperate' function :

As can be seen from the *Figure 6* this function also contains some functions. It is more reasonable to mention these functions before explaining the mechanism of data_seperate function.

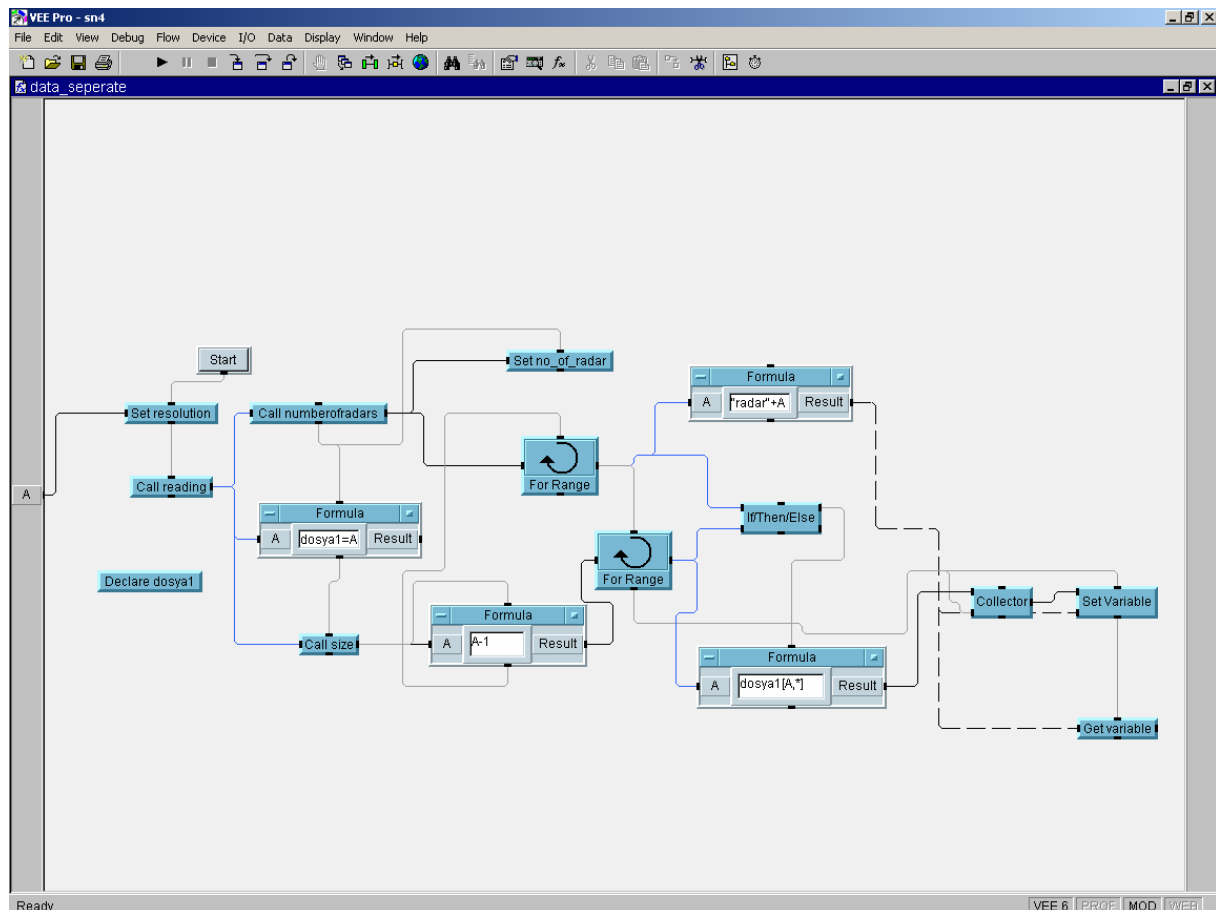


Figure 6

reading: In this function using the “from file” object the pwd file which shows the statistical results of RADSIM can be accessed. In other words the pwd file is read from computer’s disk drives. After that it is available to use in the other functions of VEE. Simply “from file” object works in that way;

First with the help of ‘File Name Selection’ function the pwd file is chosen by the user as shown in Figure 7_b. After the file is chosen it is processed in ‘From File’ function. If this function’s task is examined detailly; first EXECUTE REWIND transaction moves read pointer to beginning of file. This does not affect data. Then since the first line is composed of characters, it can be read as a string. But in the future this line will not be useful, because it contains only the names of the variables. This is the reason why I use READ NULL STR transaction. Null variable is a special VEE variable. The data read into the **null** variable is simply thrown away. Finally with READ TEXT x REAL 64 ARRAY *,8 transaction the

numerical variables are read as a two dimensional array in real numbers format. As in the pwd file the number of columns is standart, 8, and the number of rows changes the array is defined as “ARRAY *,8”. The symbol ‘*’ means ‘from beginning to the end’.

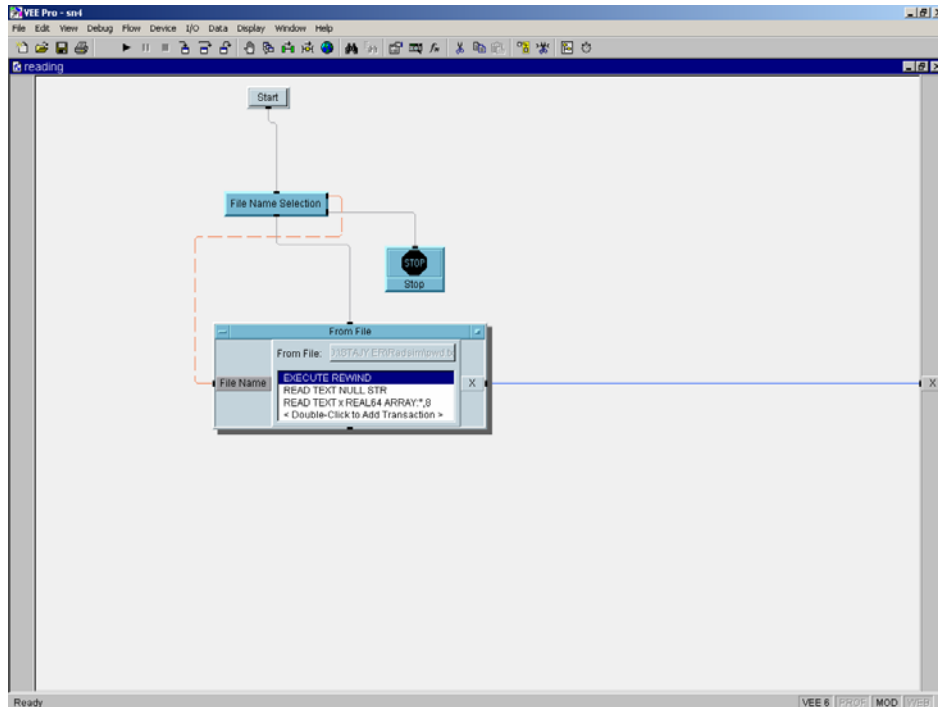


Figure 7_a

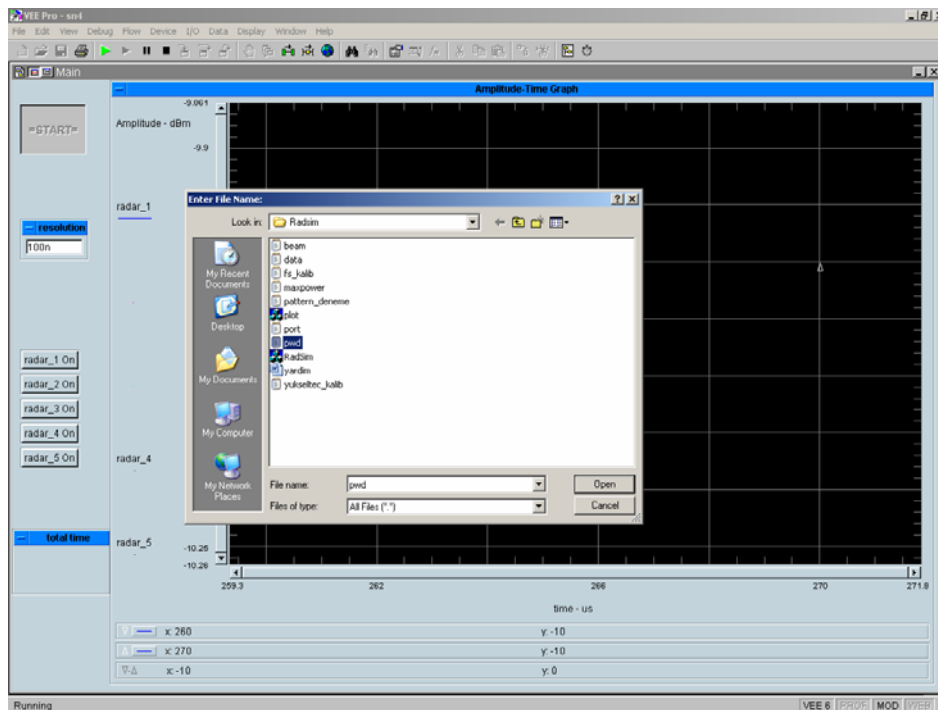


Figure 7_b

numberofradars: As clearly seen from the *Figure 8*, in this function simply the first variables of each row in the array are compared and the maximum of them is found with `max(x)` function.

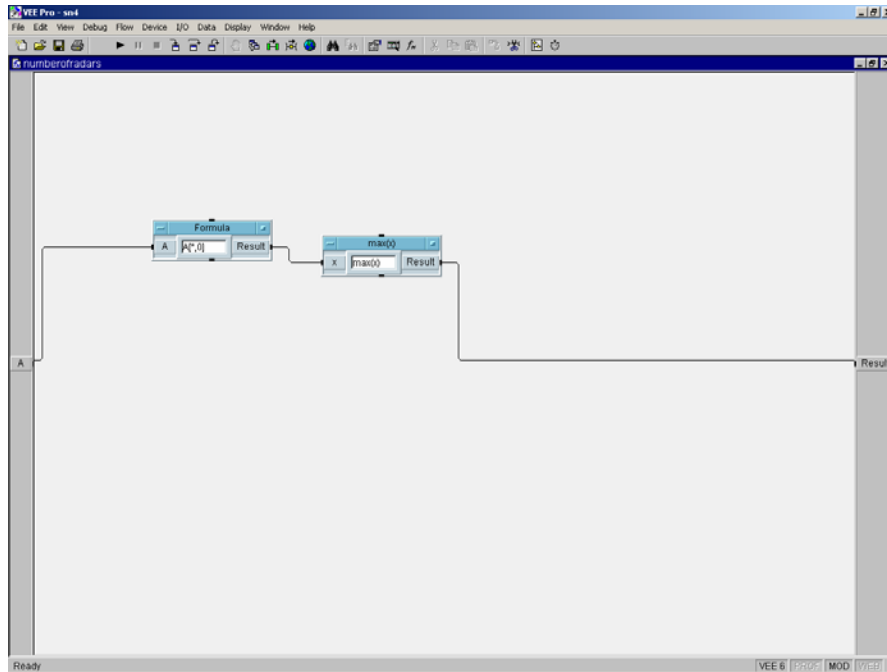


Figure 8

size: Again from the first variables of each row the total size of the array is found. 'totSize(x)' function calculates the size of this one dimensional array.

** 'max(x)' and 'totSize(x)' functions are built-in functions in VEE, these are already present in the program.

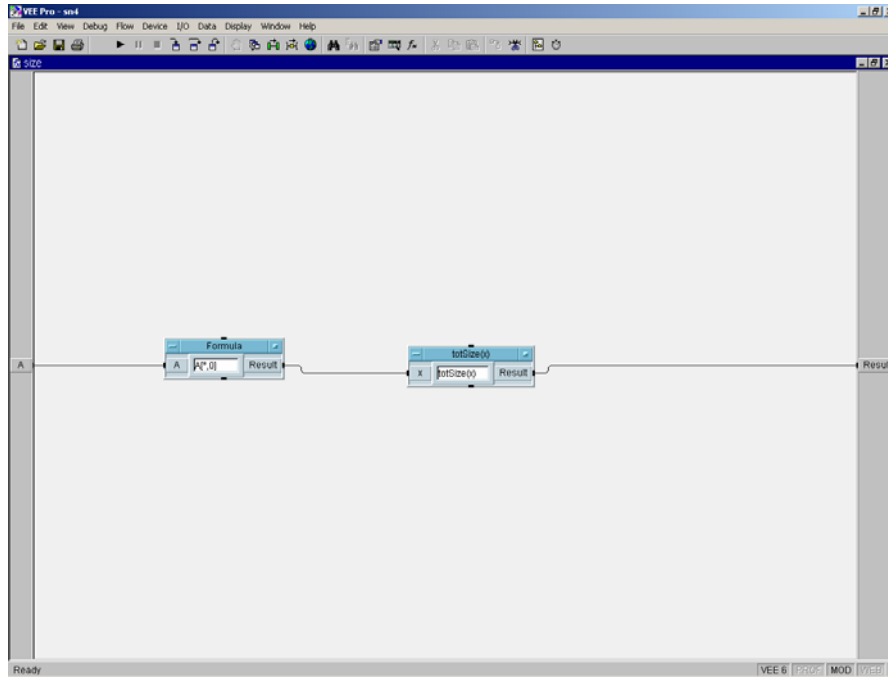


Figure 9

The mechanism of data_seperate function: First, the result of reading function, which is an array including all the necessary data related with radars, is declared as 'dosya1'. From now on it will be much more easier to reach this array. Then the radar numbers are compared in that manner, from 1 to the last radar each number is compared with the first variable of each row of 'dosya1'. If they are same this row is written in a different array with the help of collector function. Here the first for range function starts from 1. This number ,1, is sent to IfThen/Else function. On the other hand, the second for range starts from 0 (this will represent the first row of dosya1) and ends with the number which is equal to the result of size function minus 1 (this will represent the last row of dosya1). For example, first for range sends 1 to the IfThen/Else function, and 5 is sent to the function from second for range. Then a formula like $1 == \text{dosya1}[5,0]$ is produced in IfThen/Else function and if it is true the program continues to run further. If it is wrong it goes back to the second for range and increases the number to 6. Now IfThen/Else function compares 1 with the first variable of sixth row of dosya1. This control mechanism continues until the last row. Then the first row increases its data to 2 and the second for range again starts to sent numbers from 0 to the number corresponds to the last row. On the other hand another mechanism also works. I said if IfThen/Else function gives

true as a result , the program continues to run further. But, what will happen in the next steps? Actually this result ensures that this row is sent to the collector . The collector function combines all of the received data containers into an array. It continues to collect the data until its XEQ pin is pinged. It is clearly seen in *Figure 6* that this pin is pinged when the second for range completes its work. That is for example for the first radar all rows of dosya1 are considered and all the rows whose first variable is 1 are collected an array is produced which only contains the data related with radar1. Finally this array is named with 'radar1' with the help of formula box at the upper side of *Figure 6*. At the end all radars have their own arrays which contains only the data of themselves.

** Also in *Figure 6* a box called 'set resolution' is seen. This allows user to choose the resolution whatever he wants. In the main function the user determines the resolution and then in data_separate function this variable is set and named 'resolution'. That is after that when 'resolution' is seen in a formula, this corresponds to the number which the user determines.

**In *Figure 6* the result of numberofradars is also set to 'no_of_radar'. This will also helpful to design the program, because from now on 'no_of_radar' corresponds the last radar's number.

- **4.2.b. 'b' function:**

Figure 10 shows that 'b' function acts like a bridge. It first calls the no_of_radar variable ,that is, first learns how many radars exists. Then it calls each radars array and sends them to 'a' function respectively. However, it is important that first a radar's data is sent to 'a' function, then after 'a function' has completed its work with this array, next radar's data is sent to 'a' function.

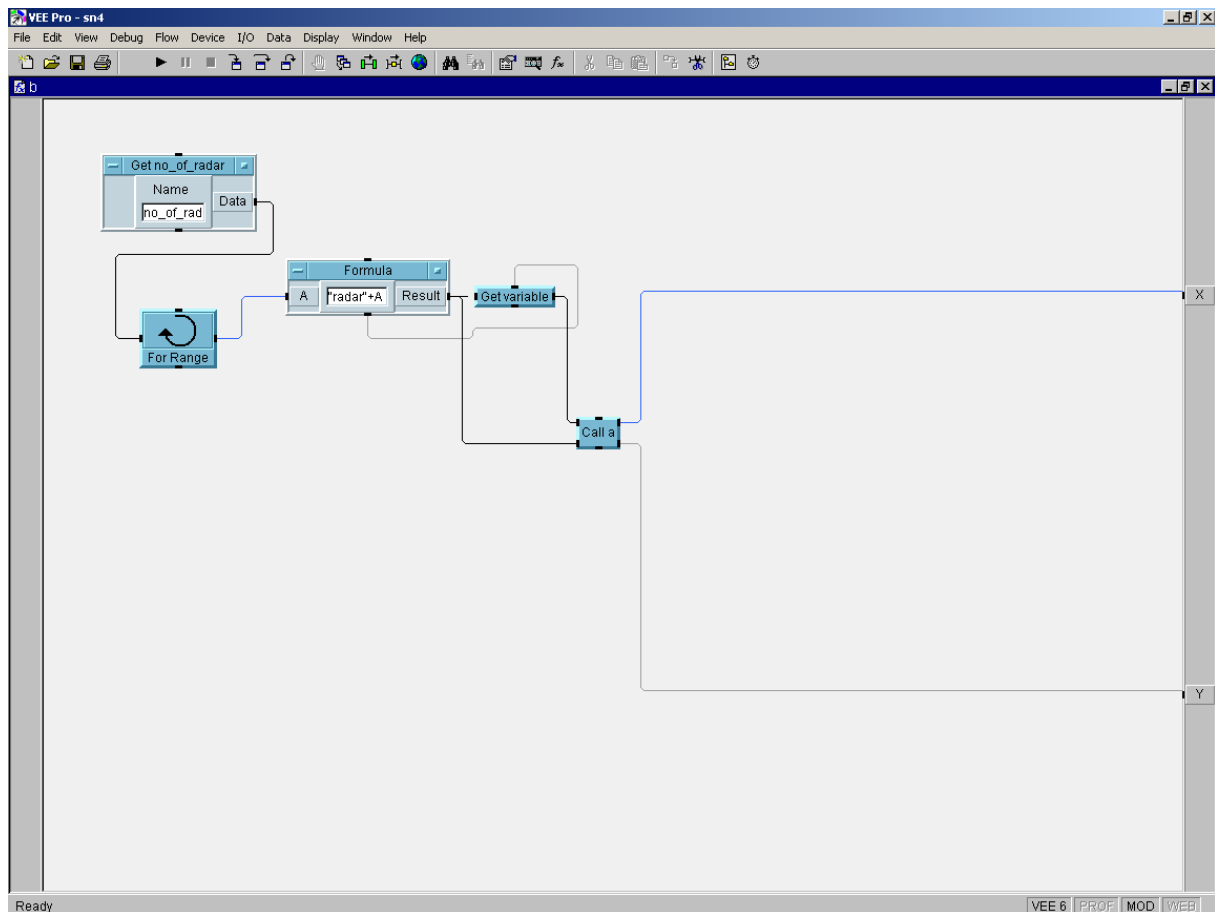


Figure 10

a: Actually this function is a control mechanism. RADSIM can create some radars that are too long. However this is a big problem for VEE. Later in this report it will be explained that VEE draws the radars' graphs by plotting points, but even in a small pulse lots of points are needed. Since these points are plotted with the help of for range function, it will take so much time for VEE to run lots of for loops. Therefore, this 'a' function controls every radar's array. If the size of the array is larger than 500, it cuts the rest of array and as can be seen in the main function when it cuts the array, a message box is activated. This box satisfies that in the panel view a warning message appears and informs the user about the lack of some pulses as in **Figure 12** .

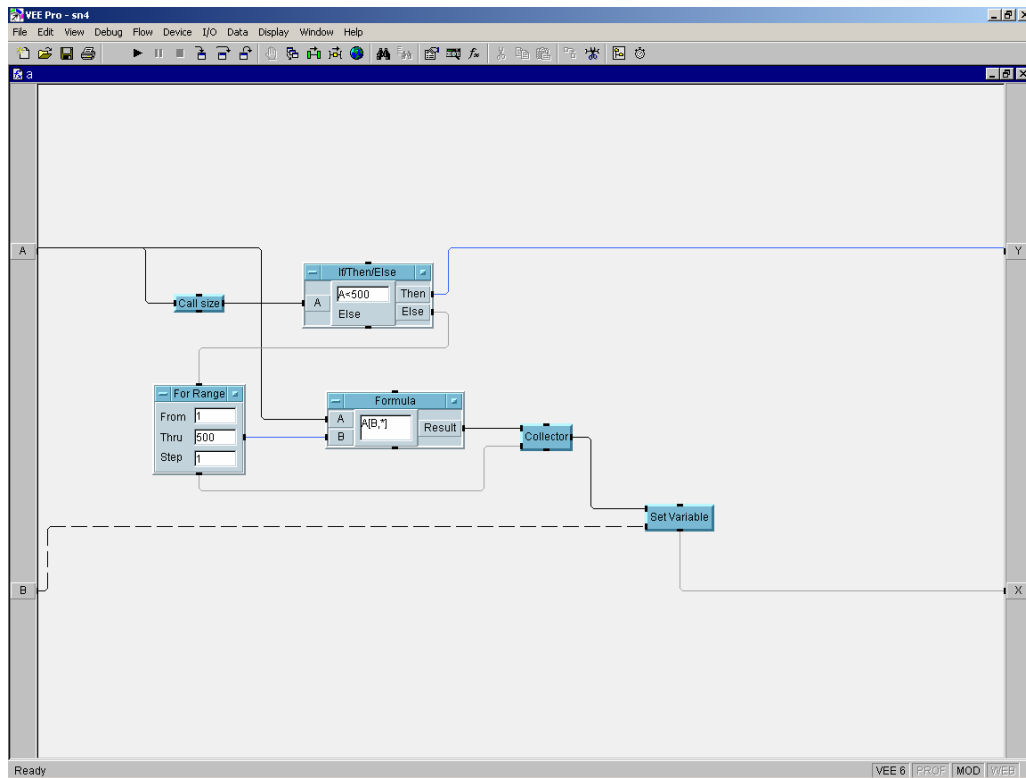


Figure 11

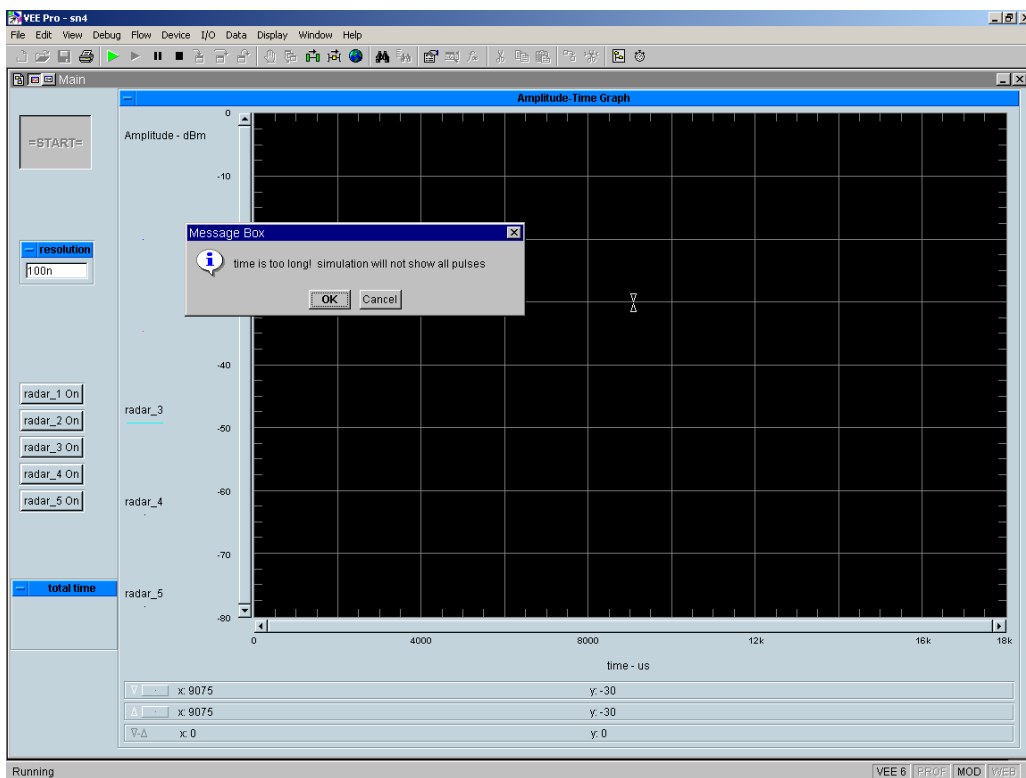


Figure 12

- **4.2.c. 'statistic' function:**

Simply this function determines the points which will produce the radar's graphical representation. As can be seen from *Figure 13*, first the function calls the array which includes the first radar's data and sends it to three different function, simulation, first and last (here another function, size2, also exists, but it is only a collateral function). In general these functions specifies the position that each point places. Finally these points are combined in a function called concatenator and puts in a variable 'radar1n'. These operations take place for all radars so that all radars have now the data sufficient to draw the graphs.

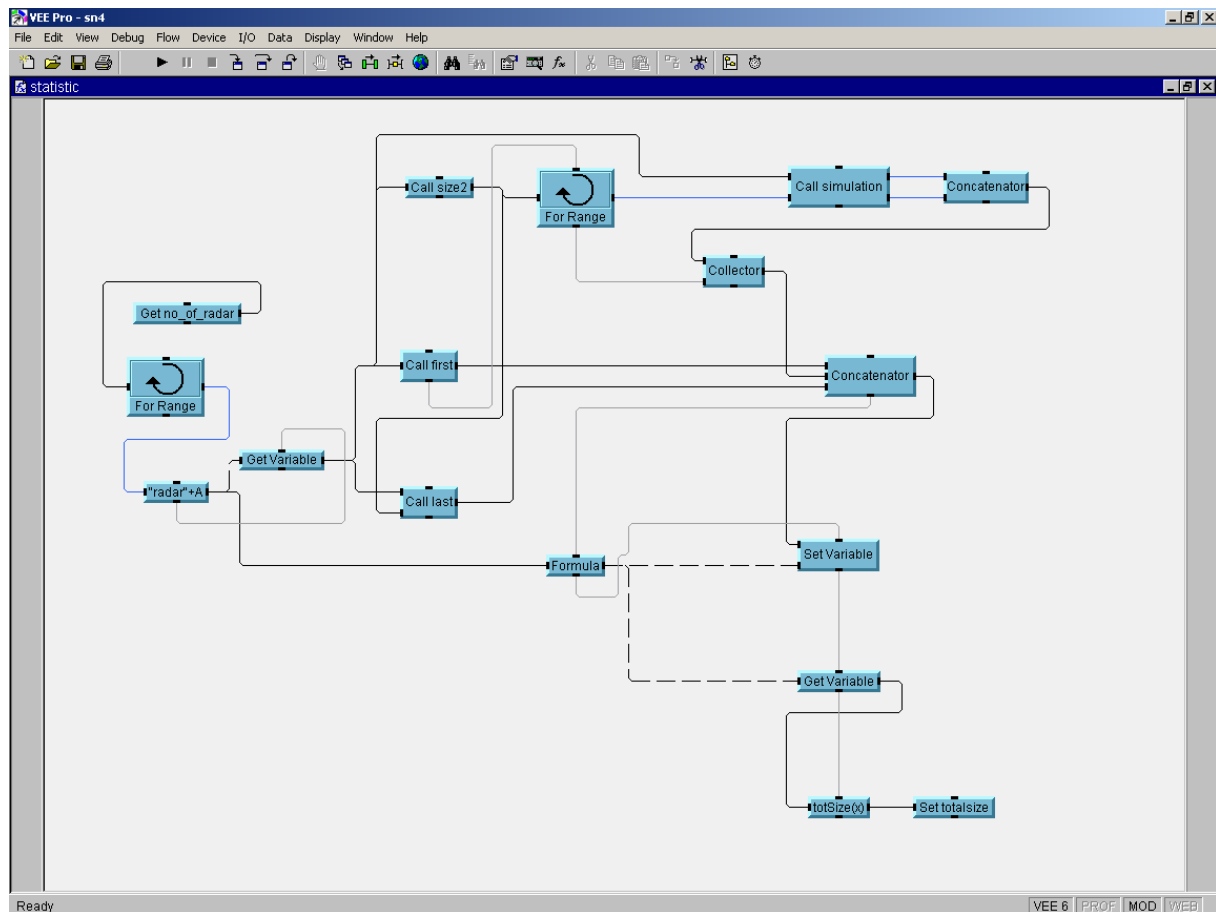


Figure 13

Now before the main three functions in the statistic function are detaily explained, I would like to mention ‘size2’ function which is a very useful function for others.

size2: In the statistic function, the first and the last pulses are examined independently of the other pulses. (The reason of this will be explained in the first and last functions.) Therefore when considering the size of an array of a radar, two rows should not be included. So ‘size2’ function is another function that calculates the number which is equal to the result of size function minus two as in *Figure 14*.

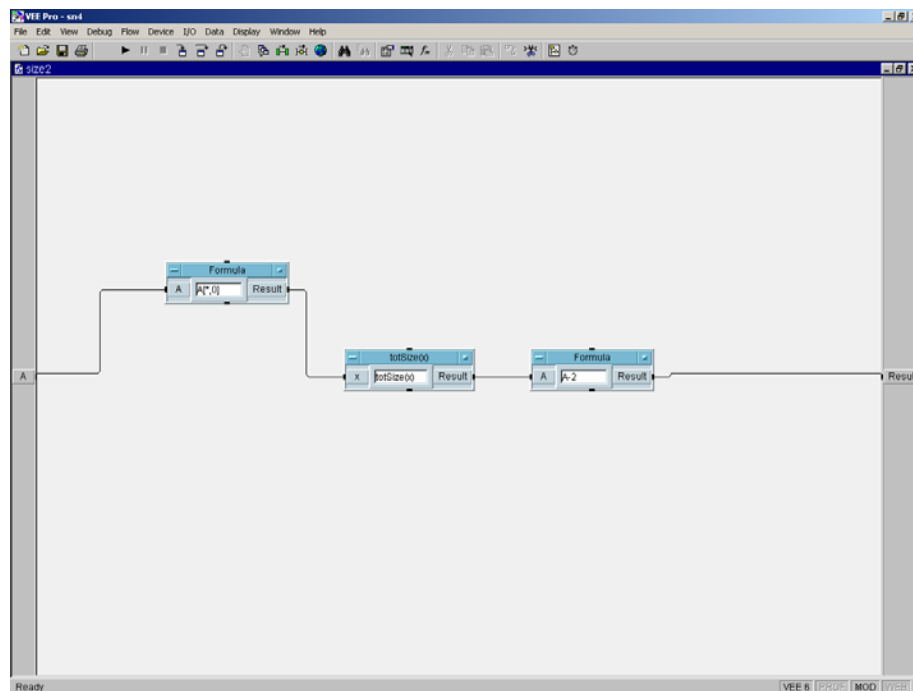


Figure 14

**Since a radar’s pulses are composed of straight lines, it would be better to understand a graphing mechanism for a small length of line before explaining the graphing a radar’s pulses.

For different resolutions, the number of points plotted to compose the line will also differ. For example for the resolution of 0.5usec a pulse which continues for 10usec in the

magnitude of -10dBm, totally 21 points should be plotted on the -10 axis. This can be easily found with the formula $[(10\mu\text{sec}/0.5\mu\text{sec})+1]$. When these points are represented as an array of one dimension, the array should be like this;

$[-10,-10,-10,\dots,-10,-10]$ (there should be 21 '-10's)

So for every pulses if an array like this can be created, composing all the arrays the whole points of a radar can be plotted so that the radar can be represented graphically. That is the way how things work in statistic function, first, simulation and last functions create one dimensional arrays and concatenator combines them. Finally long one dimensional array will have been produced whose names are like 'radar1n, radar2n....'.

If the resolution is 1 usec the result will be 11 as seen in *Figure 15_a* and *Figure 15_b*. However the requirement of my work is that the resolution should be 100nsec, that is, to represent 10usec 101 points should be plotted.

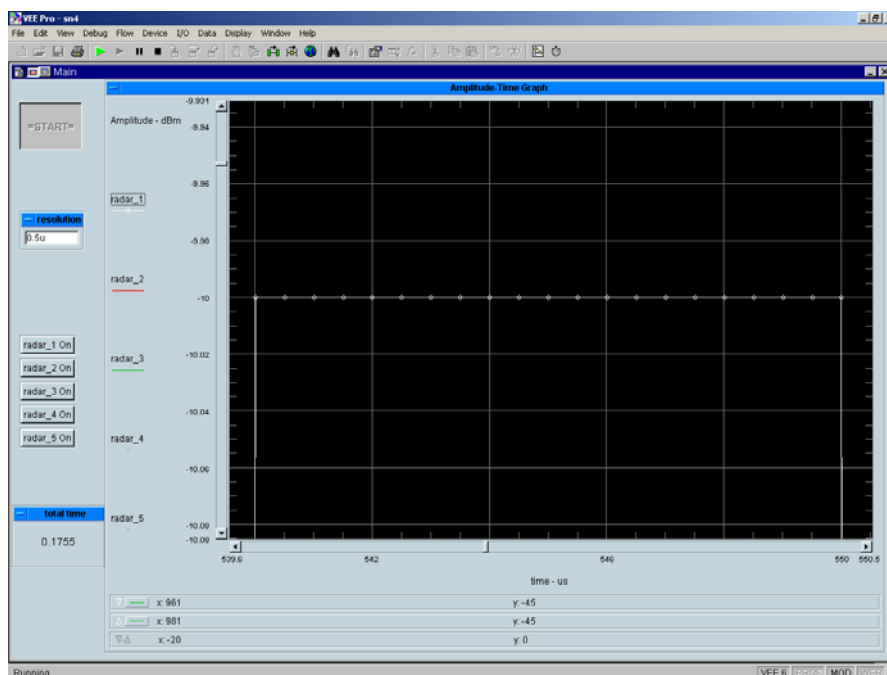


Figure 15_a

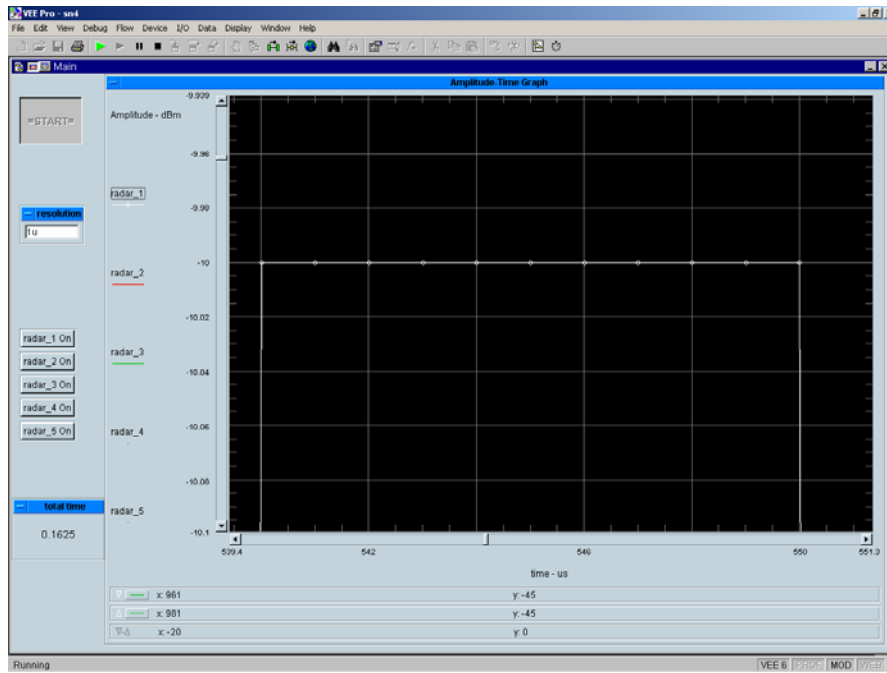


Figure 15_b

first: The main purpose of this function is to draw the first pulse of a radar. First, this function controls whether the first pulse starts at $t=0$ or not. For example, for the first radar if the variable corresponding to `radar1[0,4]` which shows the first pulse's starting time is 0, then first the points on the pulse are written on an array, second the points representing 'OFF' sign are written on a different array and finally they will be combined with the help of concatenator. These are shown at the upper side of *Figure 16*. In the figure a function called 'Alloc Real64' is used. This function creates an array with a size that can be adjusted. On the other hand, if `radar1[0,4]` is not 0, that means the first pulse will be later. Therefore first an array of '-80's (it is the value when the sign is off), second an array composed of the magnitude values of the pulse, and finally again '-80's (these should continue until the second pulse) should be produced. Then the concatenator will combine them with an order. First the input came from A pin, second the input from B, and finally the input from C are taken and written on a different array.

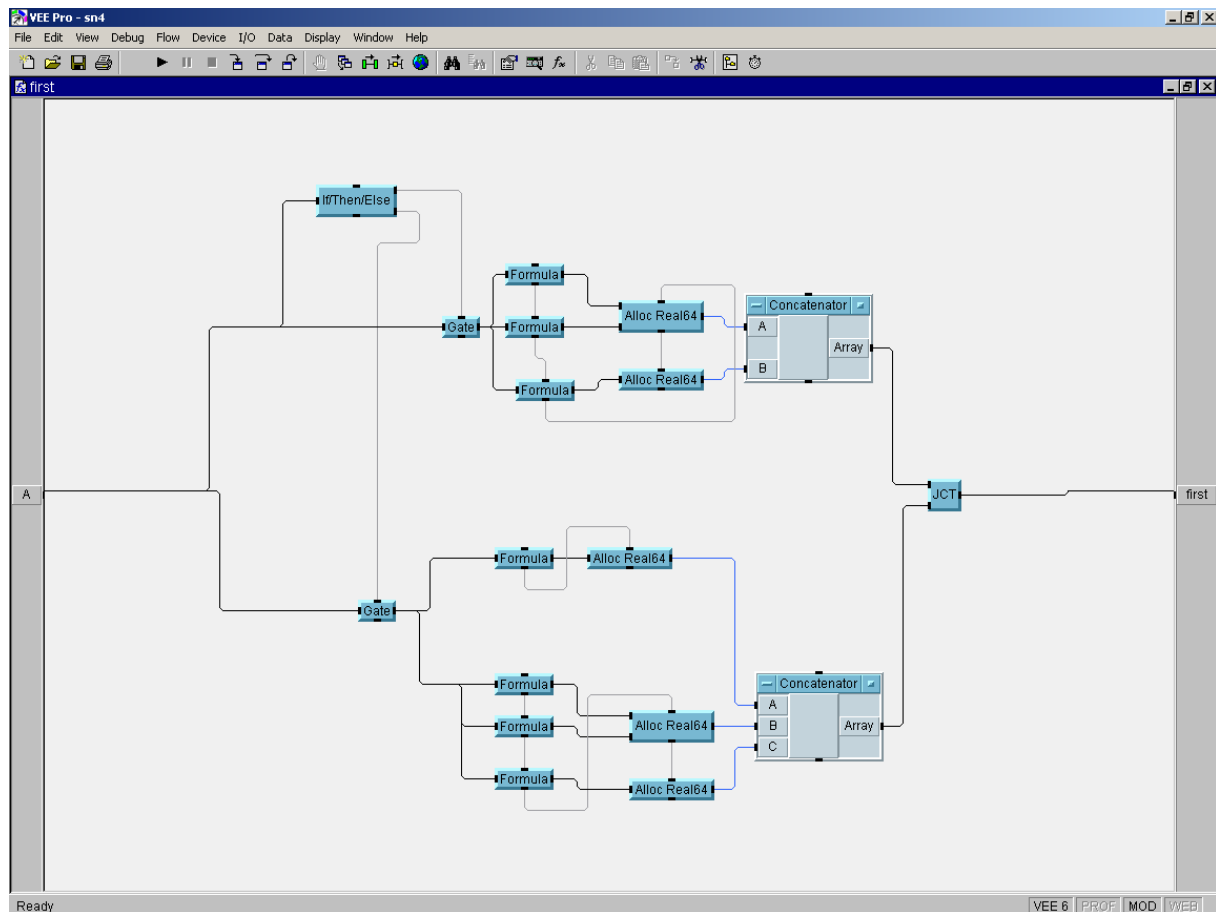


Figure 16

simulation: In the simulation function, starting from the second pulse up to the last pulse all points are written on an array. Finally for example for a radar which has pulses at -15 dBm an array occurs like that [-15,-15....-15,-80,-80.....-80,-15,-15....-15,.....-80,-80].

last: In this function, the last pulses of radars are drawn by again plotting the necessary points. As can be seen from **Figure 17** at the end of the array the number '-80' is added, this is because of completing the pulse. When this is added the pulse will have been done.

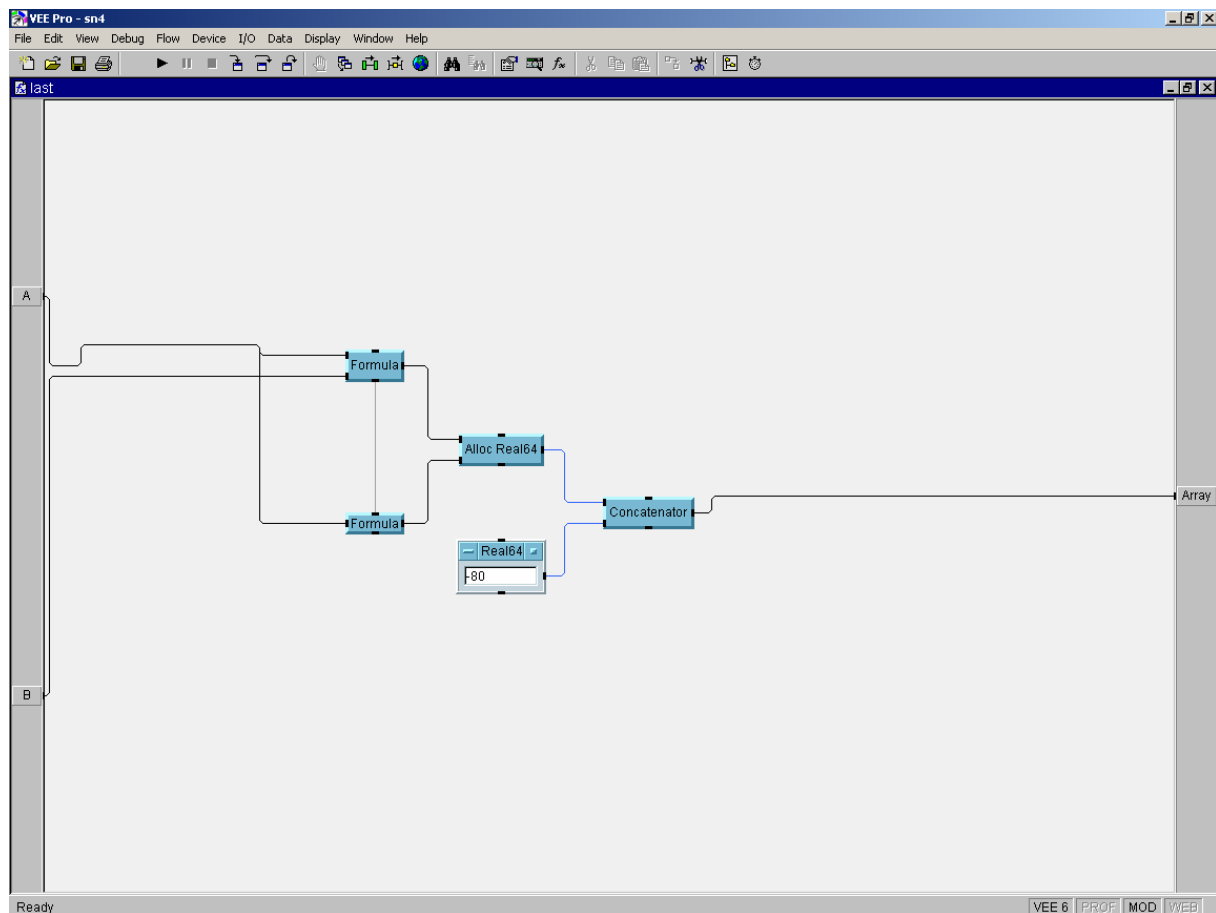


Figure 17

- **4.2.d. ‘setting’ function:**

Actually when this program has completed, maximum 5 radars can be represented graphically. This is the desirable situation. Therefore all the setting relating with graphing will be done for 5 radars. However what if the user defines less than 5 radars? Then since the data of the undefined radars is empty, the program will give an error message. That is, it does not work. To prevent this the setting function has been composed. In this function an array of size 10 and includes only ‘0’s is set to the radars which has not been defined with RADSIM by the user. Actually the arrays created for undefined radars are not really important, because

when they are plotted, the settings will be adjusted in such a manner that these undefined radars will not appear in the graph.

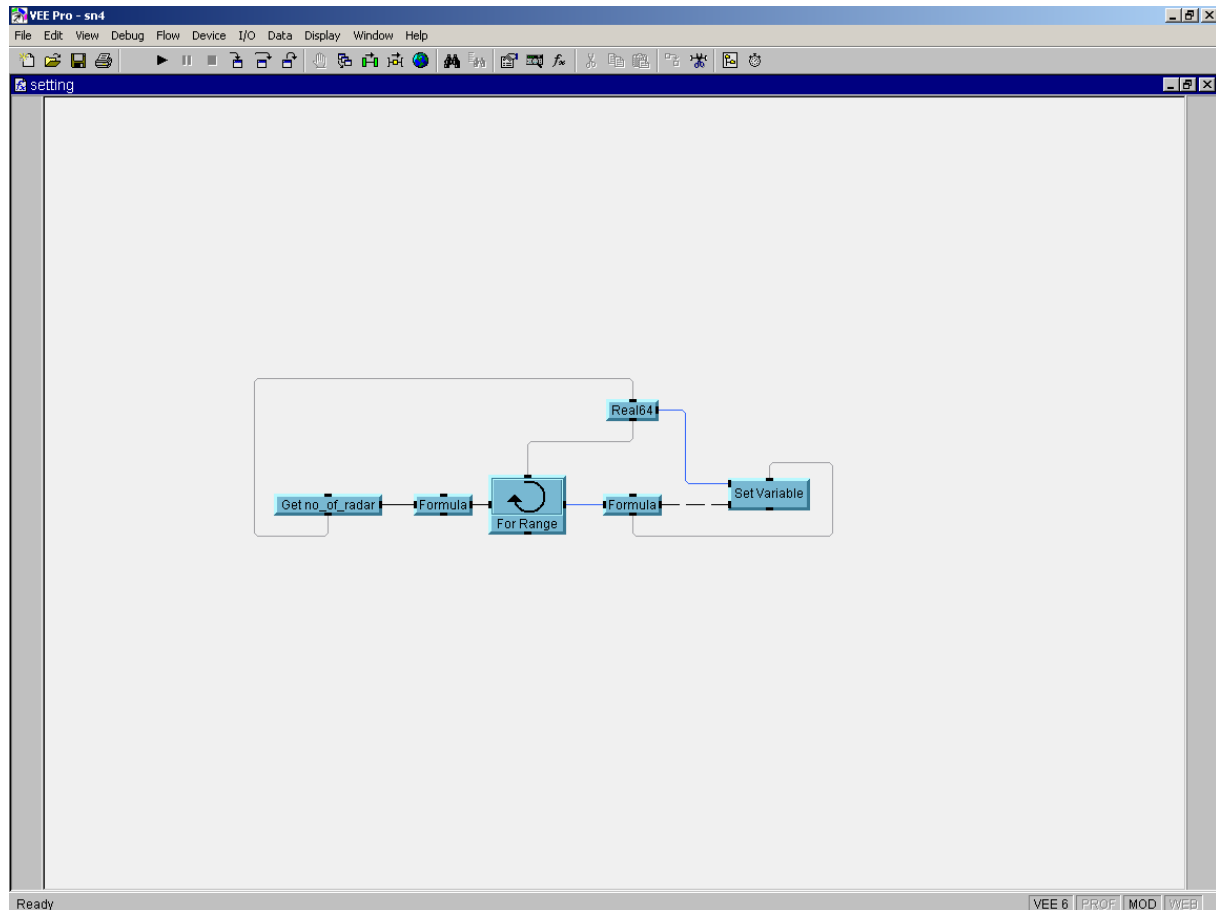


Figure 18

- **4.2.e. 'sizes_n' function:**

This function is also a precaution for future error messages like setting function. Since sizes of the arrays 'radar1n,radar2n,...' are all different, the program will not work and give an error. Therefore all the radars' sizes should be equalized. To do that first the sizes of radars defined in RADSIM should be known.(It is not necessary for undefined radars, because we have already set their sizes as 10.) Here this function calculates the sizes of the long arrays

- **4.2.f. ‘min_size’ function:**

31

size. Actually, this function is responsible for finding the size of the shortest array, that is, the radar's data with the minimum variables and setting it to a variable called min_of_size.

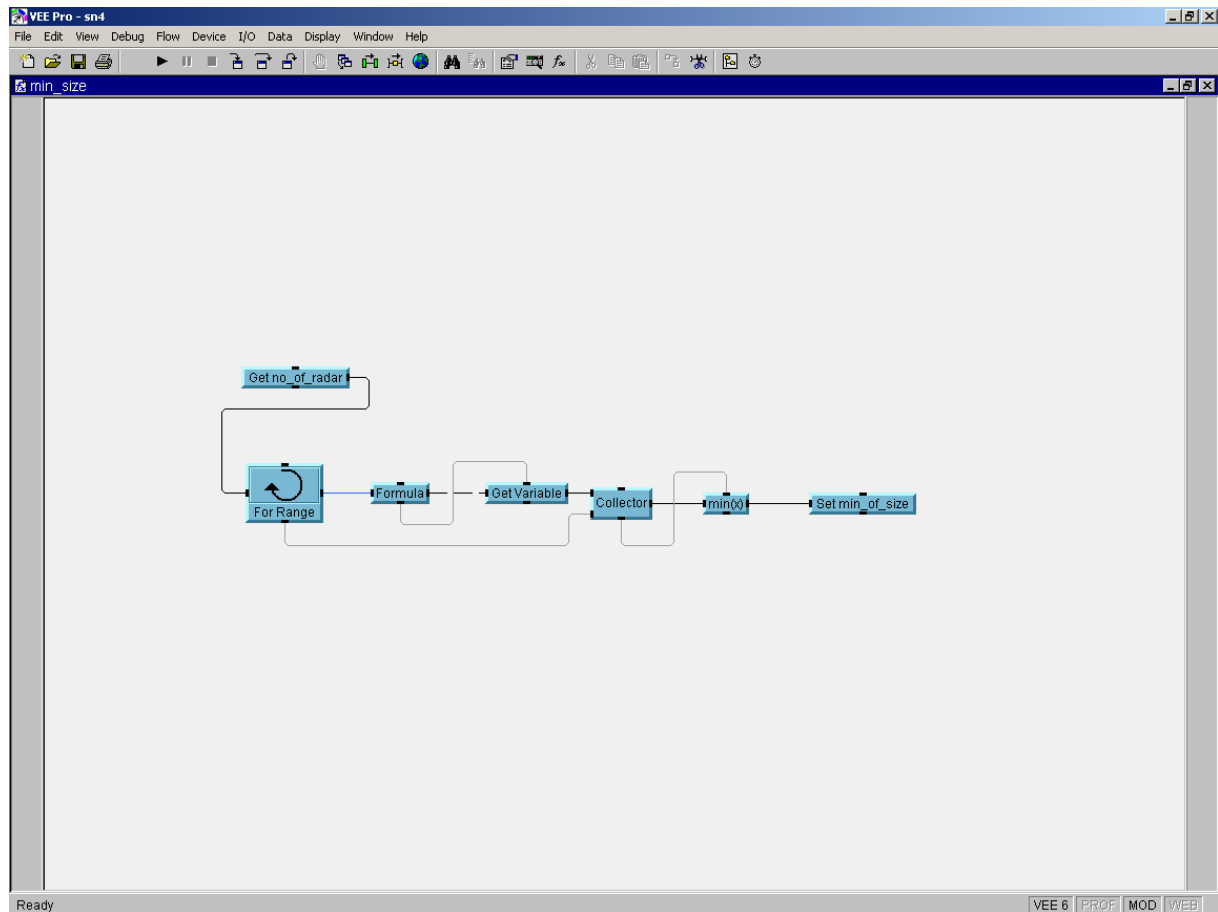


Figure 20

- **4.2.g. 'graph' function:**

In this function, all radars are called and prepared for graphing, that is, their sizes are equalized in the subfunction called 'making_same_size'. On the other hand another array should be produced. This array which will be set as 'radars' can be thought as an initialization. It can be seen in *Figure 4* (the main function) 'the radars' variable is directly sent to the amplitude-time graph. Actually if it is required for the graph for setting the size of the figures which will be plotted to the graph. For this purpose an array is produced in the graph function by Alloc Real64 with a size of the shortest radar data's size.

In Figure 21 it is seen that all radar's data are sent to 'making_same_size' function separately. The reason behind this is that after their sizes are equalized they should be set as different variables and these new variables should be sent to amplitude-time graph separately. Because the graph box takes all the datas and then plots them, rather than plotting one data after the other.

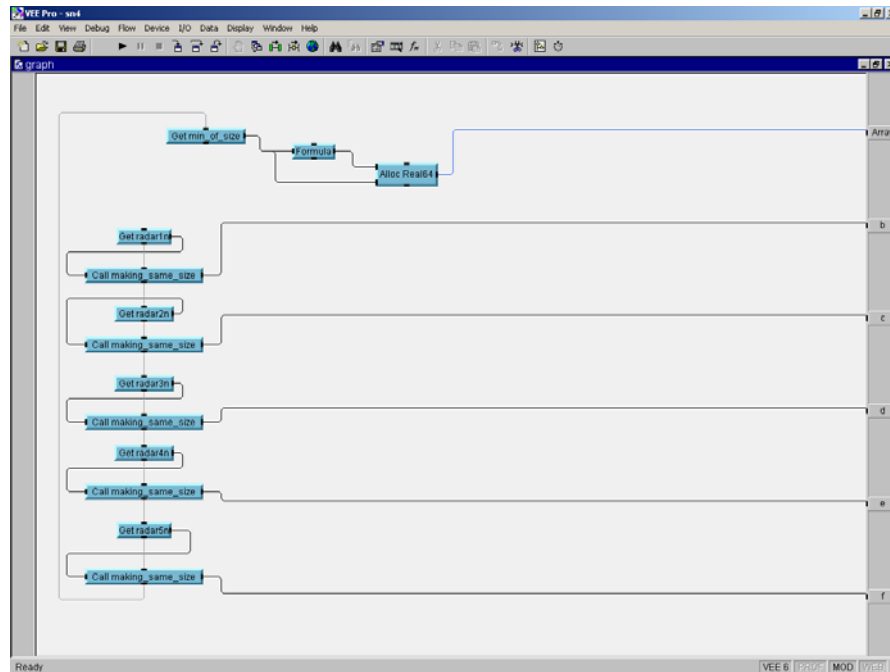


Figure 21

making same size: After calculating the size of the array came to the function in the 'sizee' subfunction it is sent to IfThen/Else function for comparison. After that this function can be explained in three parts;

1. When the size of radar's data is equal to min_of_size, that is, this array is the shortest array, then it is directly sent to junction without any changes. Junction allows data outputs to be combined so they can be connected to one data input. every time an input is pinged, that data is sent to its output. So junction will sent the data came from A pin, which is the unchanged radar data, to the output.

2. If the size of radar's data is not equal to min_of_size, then it is compared with 10 in order to understand whether this radar is defined by the RADSIM or not. When it is an undefined radar the formula box at the bottom of *Figure 22*, which has a formula $[\text{min_of_size}-10]$ in it, is activated. Finally with the help of Alloc Real64 and Concatenator function, the rest of the array will be filled with '0's. At the end we will have an array of '0's with the desired size.
3. Finally if the radar is a defined radar and the size of the radar's array is larger than min_of_size, it is shortened with a simple formula like $A[0:\text{min_of_size}-1]$. This function cuts the variables from min_of_size to the last, it only takes the variables up to min_of_size.

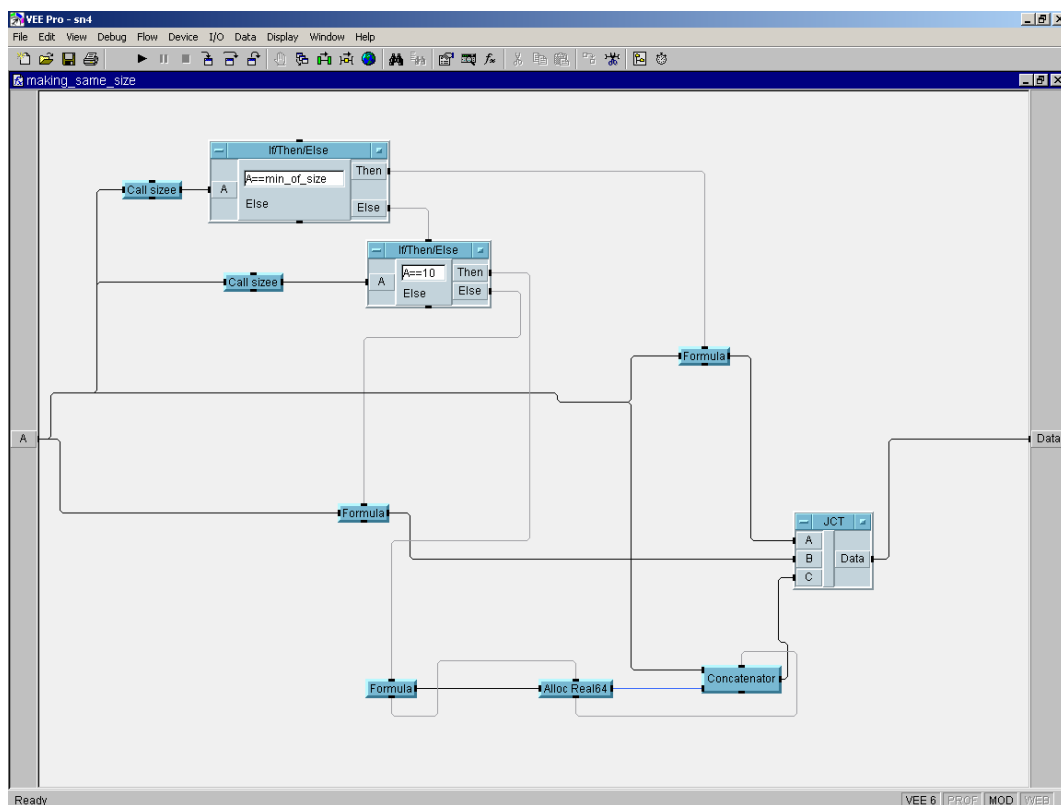


Figure 22

- **4.2.h. ‘first_setting’ function:**

This function takes the arrays of size of min_of_size for each radar and it sets them to different variables called ‘radar_1,radar_2,.....,radar_5’. In addition, this function calls a function ‘record_initialization’ and sets its results as TraceSetup.

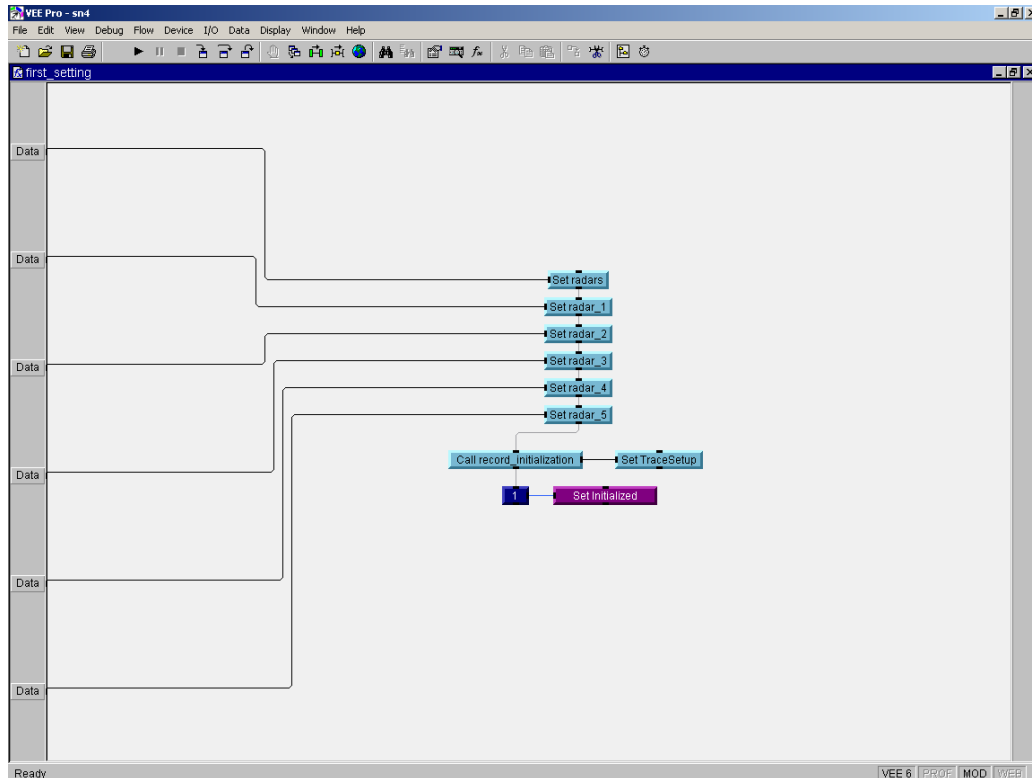


Figure 23

Record initialization: In this function an initialization record is built. This record includes radar’s numbers, names, pen color type, line type, and point type. Since there are at most 5 radars, tracenum has variables from 1 to 5. In the name field of the record, the names of radars such that ‘radar_1,radar_2,.....,radar_5’ will be written. Pen field should consist of numbers that correspond to different color. However, if the radar is not defined in RADSIM, it should not take place in the graph. So with a ‘IfThen/Else’ function the radar’s data should be examined as in *Figure 24*. If the array is composed of all ‘0’s, then it is an undefined radar. So the number corresponding to the color ‘gray’ should be written in the pen field. If it is a

defined radar, any number can be written on pen field. Here since the number is increased by 4, the radar's color will be the color corresponding the number which is equal to the radar number plus 4. (eg. the first radar's color will be blue which is the reciprocity of 5.) For line type again the number of undefined and defined radars should be different not to show undefined radars. On the other hand, point type is not so important. Here we can choose the number corresponding to no points on the line or the number corresponding to diamonds on the line, etc...

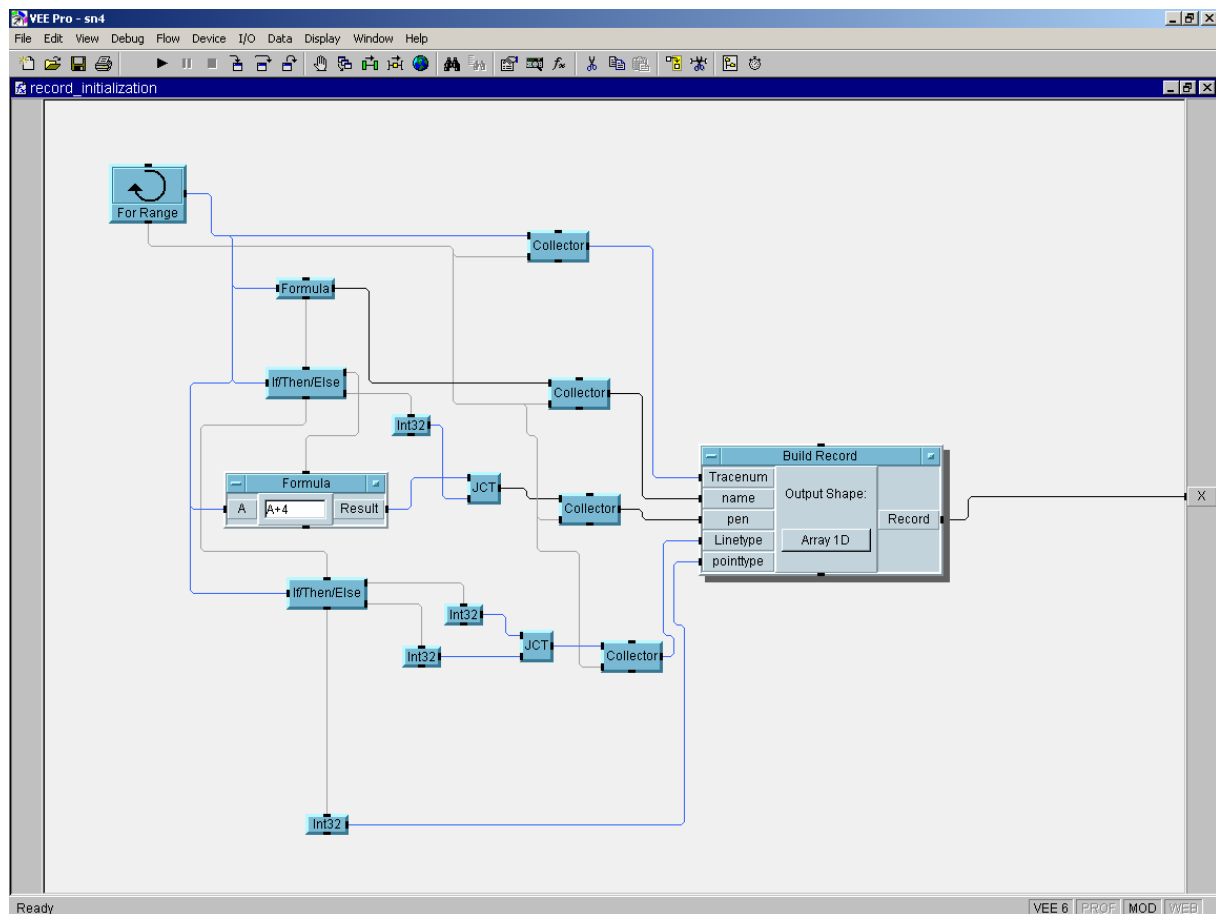


Figure 24

Now, since this record is set to a variable called 'TraceSetup', TraceSetup is a one dimensional array with a size of 5 including all the necessary variables to draw the radars' graphs. So now the designing of the program has been completed. In the main function 'TraceSetup' array is also sent to amplitude-time graph such that it identifies the radars'

colors, type of line etc... Now, when the program is started it will shows the radars defined in RADSIM graphically. For a pwd file shown in *Figure 3*, the graph will be like in *Figure 25*.

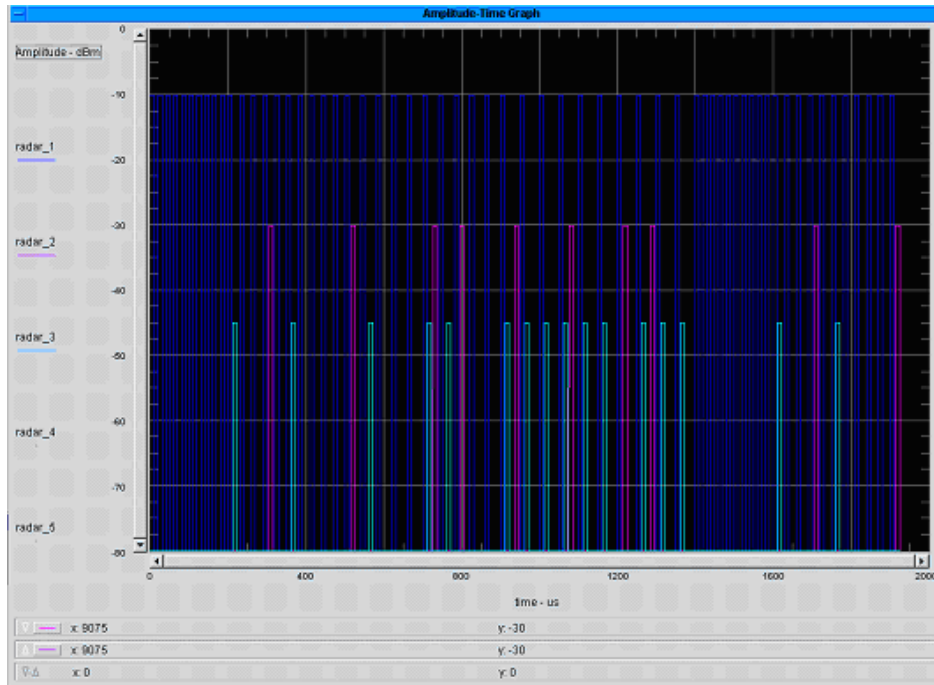


Figure 25

4.3. IMPROVEMENT OF THE PROGRAM :

Although colors of each radar are different, sometimes, especially when the pulses of the radars are so frequent, radars can be mixed. In other words the graph can be confusing. However if the user can see only one radar's graph, it would be much more convenient for him. In fact, after completing the program, I added a function called 'buttons' to the program. This function had been prepared before by the TMM-REH engineers. I have only modified it and connected it to the program I designed.

- **4.3.a. 'Buttons' function:**

In this function, for all radars a cyclic button is produced. At the beginning these are all assumed to be ON. Then when they are changed to OFF, radars will not appear in the graph as if their variables were deleted. Then if they are again changed to ON their variables are all adjusted again and set to TraceSetup's related variable. For example, if the user make radar_1 ON after making it OFF, the color of radar_1 will be changed, because earlier the color corresponding 5 is the color of radar_1, but from now on it is the color corresponding to 1(the pin in the middle of the upper 'build record' function which corresponds to pen type is connected to the number 1 in *Figure 26*).

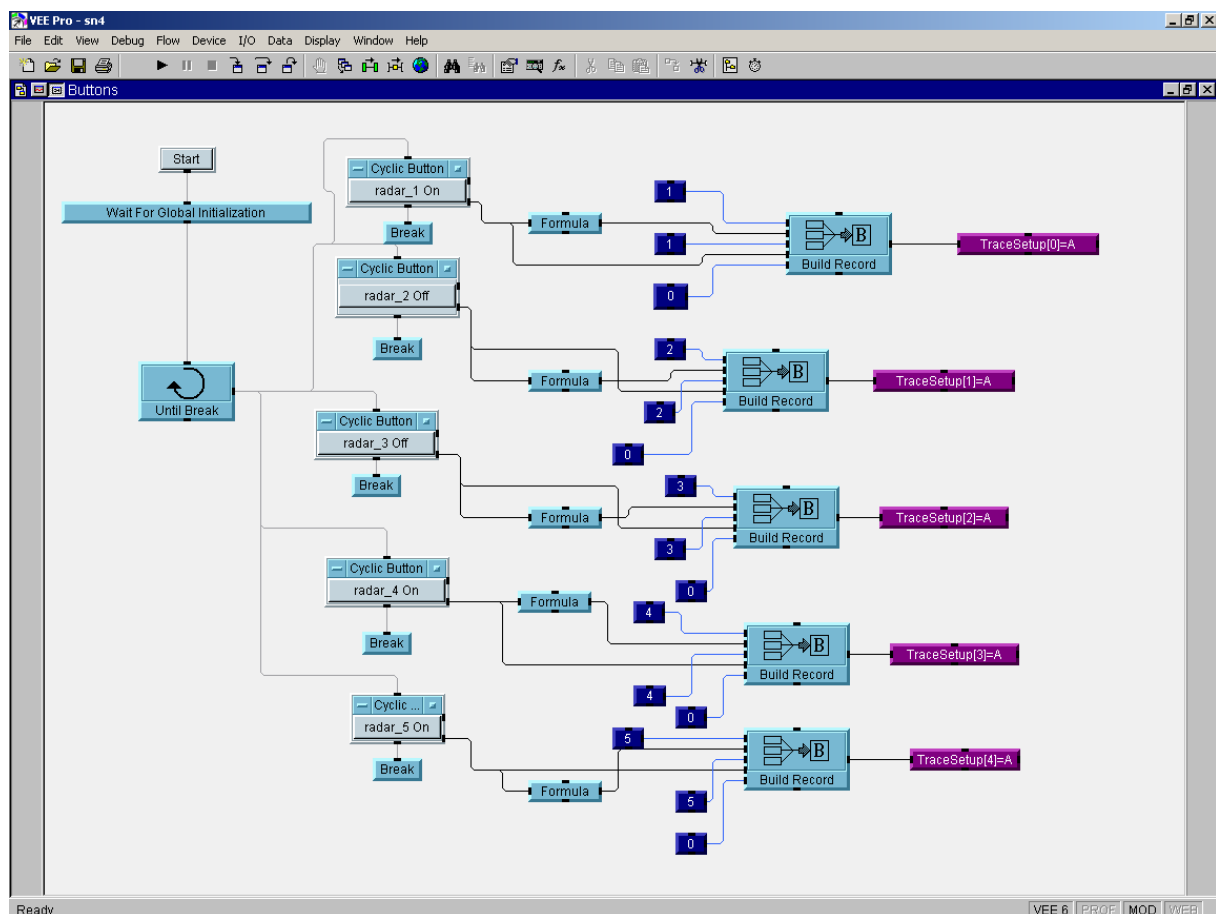


Figure 26

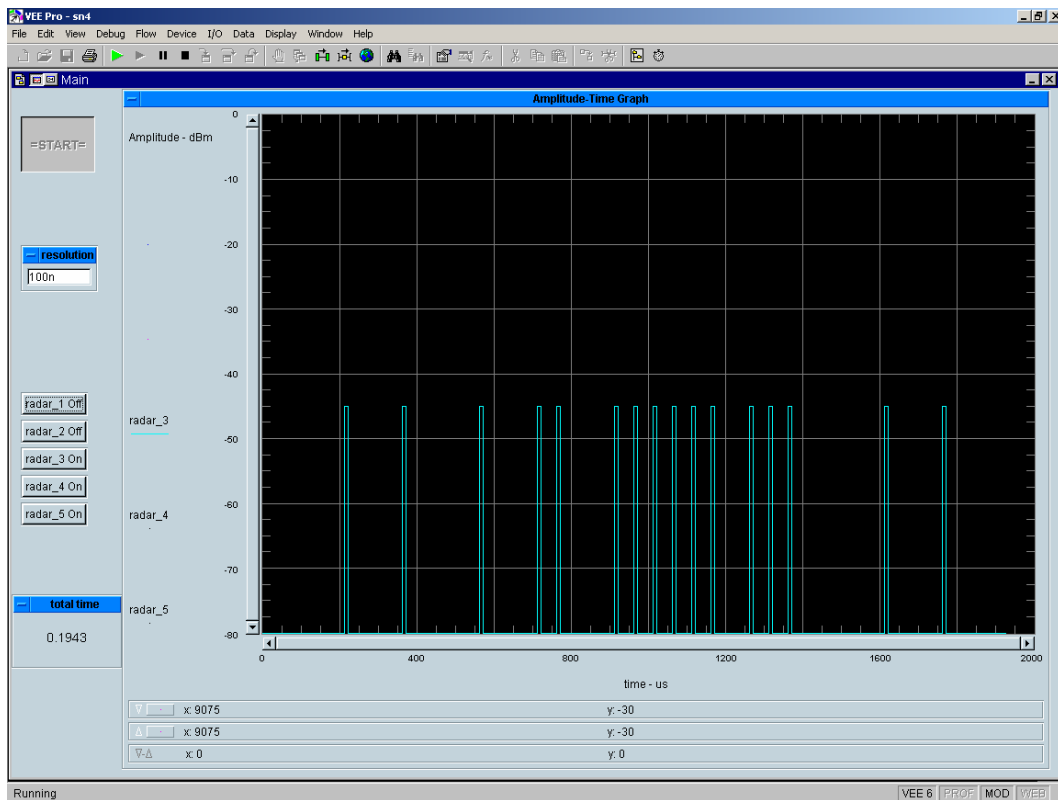


Figure 27 – when radar_1 & radar_2 are off

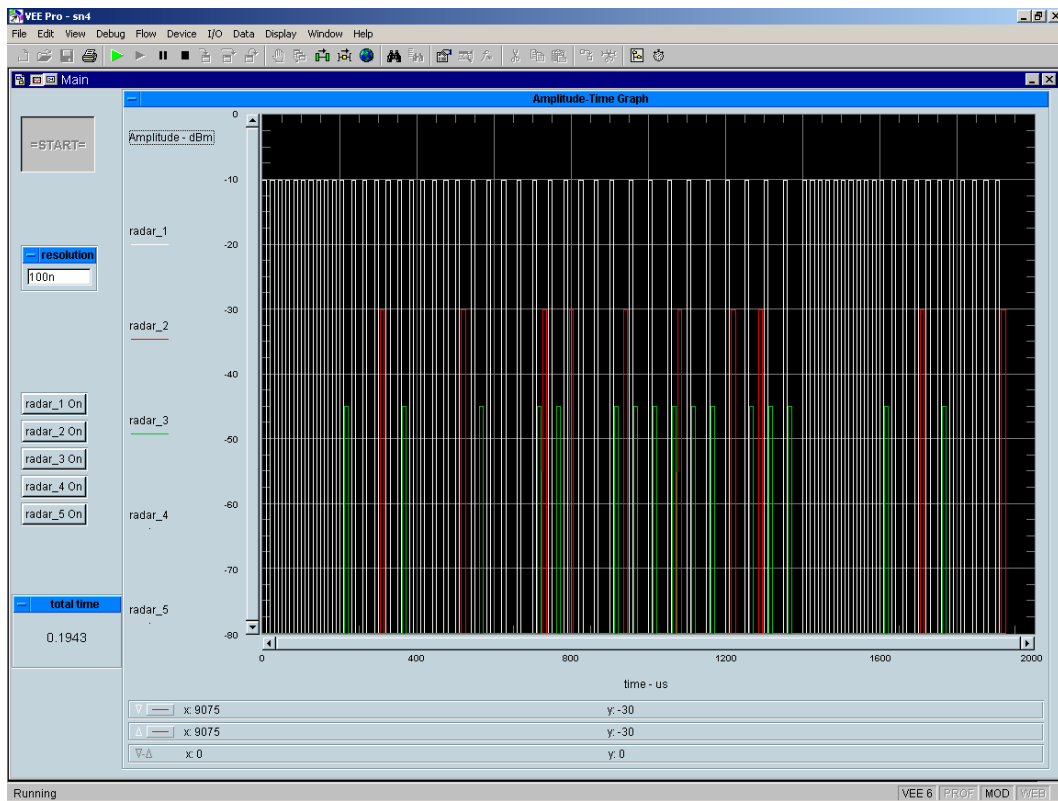


Figure 28 – when all the radars are on again

5. GRAPHING THE RESULTS OF RADSIM WITH VISUAL C++ :

The same procedure used in VEE can also be applied when Visual C++ is used for graphing; first the file should be chosen, then the data in the file should be read, since the radars' data are unordered, they should be separated, and finally with the help of already existing DLLs the graph of each radar should be plotted. However, the method will be different than VEE, because in Visual C++ the code should be written by hand and there is no predefined functions that help the programmer.

In Visual C++ the workspace contains three parts; class view is the view where the code is written, on the other hand resource view contains dialog boxes, buttons, icons, etc.. (it can be considered as visible part of the program for the user) and finally in the file view all the files that make up the application exist.

At the end when this program will have completed, there will be a dialog window named 'new exercise' like in **Figure 29** including some buttons, a scroll bar and some static texts which simply show the written texts or drawn pictures, in fact they simply paint the characters on the screen.

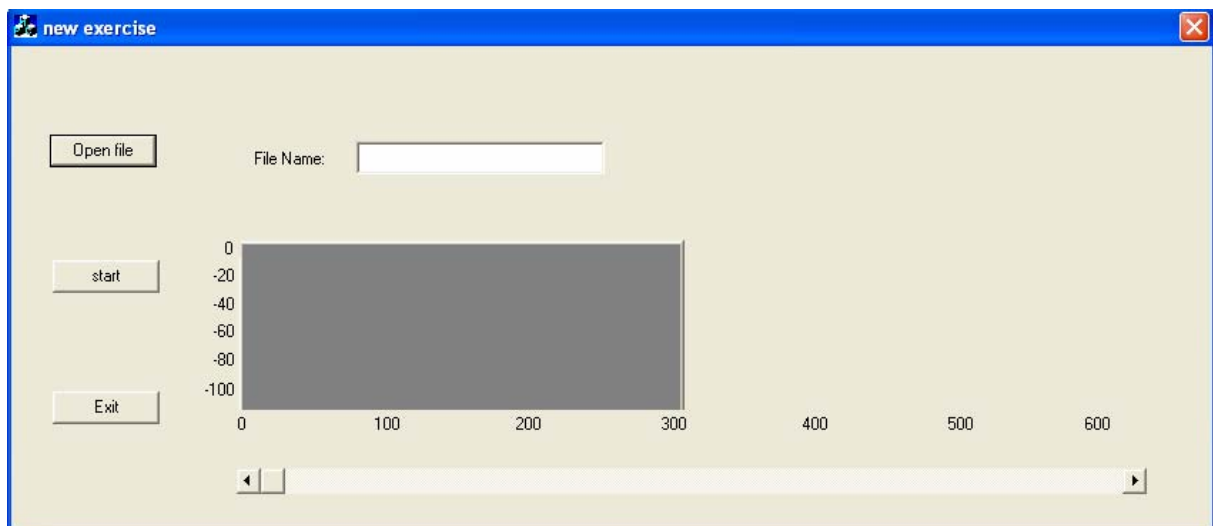


Figure 29

It is convenient to understand how the program works, if each control in the dialog window and their tasks are understood. For this reason I will explain each button separately in the report. However since these buttons control static texts and these texts are only responsible for showing the data on the screen, I will explain these texts under the related button's heading.

5.1. 'OPEN FILE' BUTTON :

First, a button called 'open file' has been added to the dialog window of the program and its ID is chosen as IDC_FILEOPEN. Then, in ClassWizard the activation of the button is done choosing BN_CLICKED, which means that when this button is clicked one times it is activated, and adding the function to open the file as in *Figure 30*.

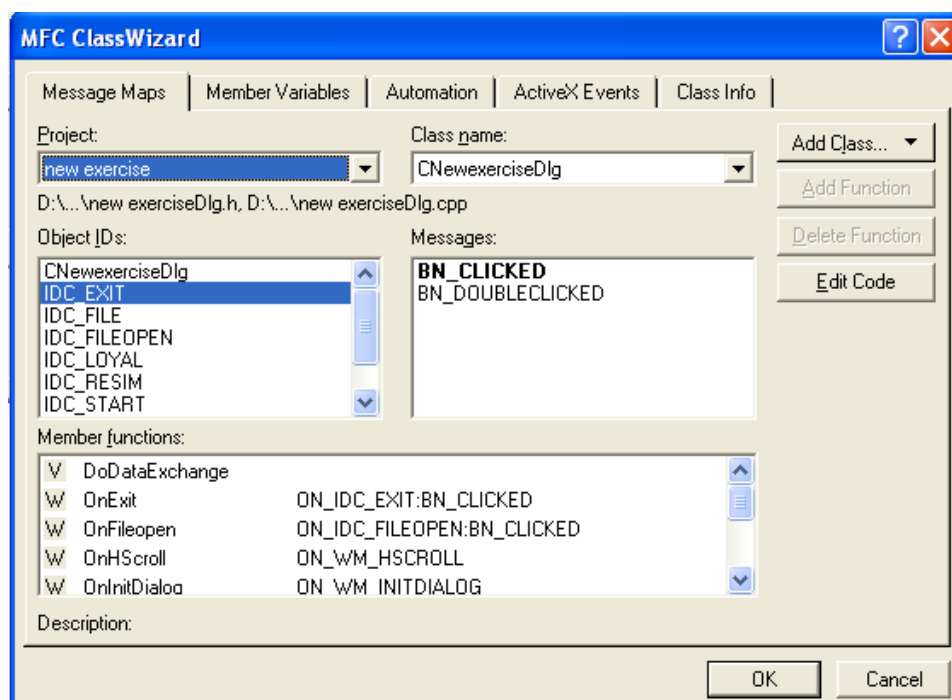


Figure 30

When a function is created, inside the function some codes can be written. For example for this function a code which is needed to open the file page should be written like shown in *Figure 31*. In fact this open file button also controls a static text that shows the name of the file chosen. In *Figure 31*, the code written also includes this event.

```

void CNewexerciseDlg::OnFileopen()
{
    // TODO: Add your control notification handler code here
    CFileDialog m_ldFile(TRUE);

    // Show the File open dialog and capture the result
    if(m_ldFile.DoModal() == IDOK)
    {
        // Get the filename selected
        m_sResults = m_ldFile.GetFileName();

        // Update the dialog
        UpdateData(FALSE);
    }
}

```

Figure 31

****In the code a variable called ‘m_sResults’ exists. This is a variable of the static text defined as a string.**

If the code in **Figure 31** is examined, it is seen that when the ‘open’ dialog box appears on the screen like shown in **Figure 32**, and when the file is chosen this variable is changed to the chosen file’s name. In my project, since the results of RADSIM is saved as a pwd file, the user should choose the pwd file which stays under the Radsim file.

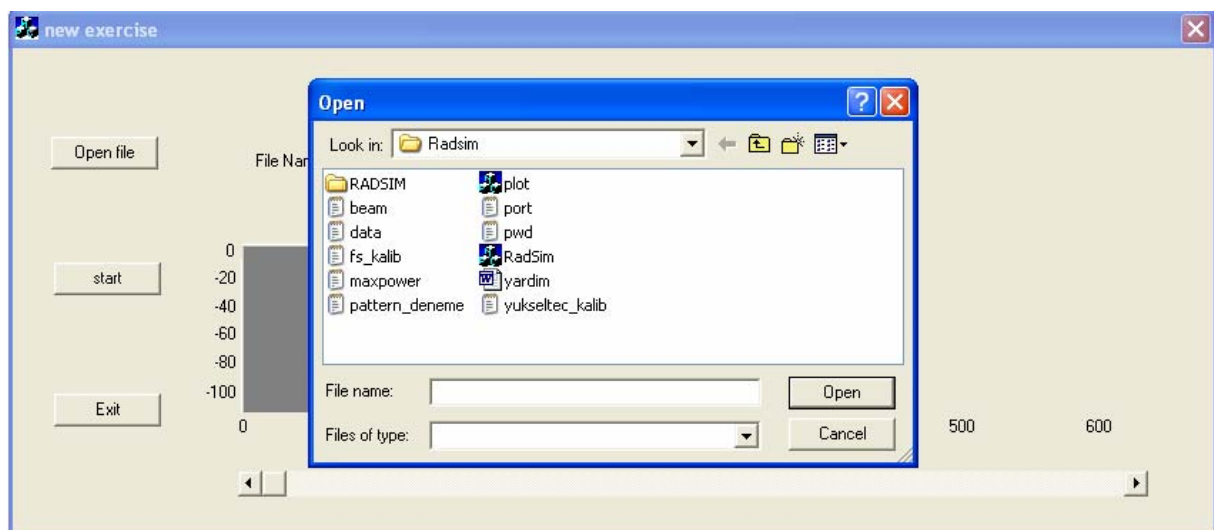


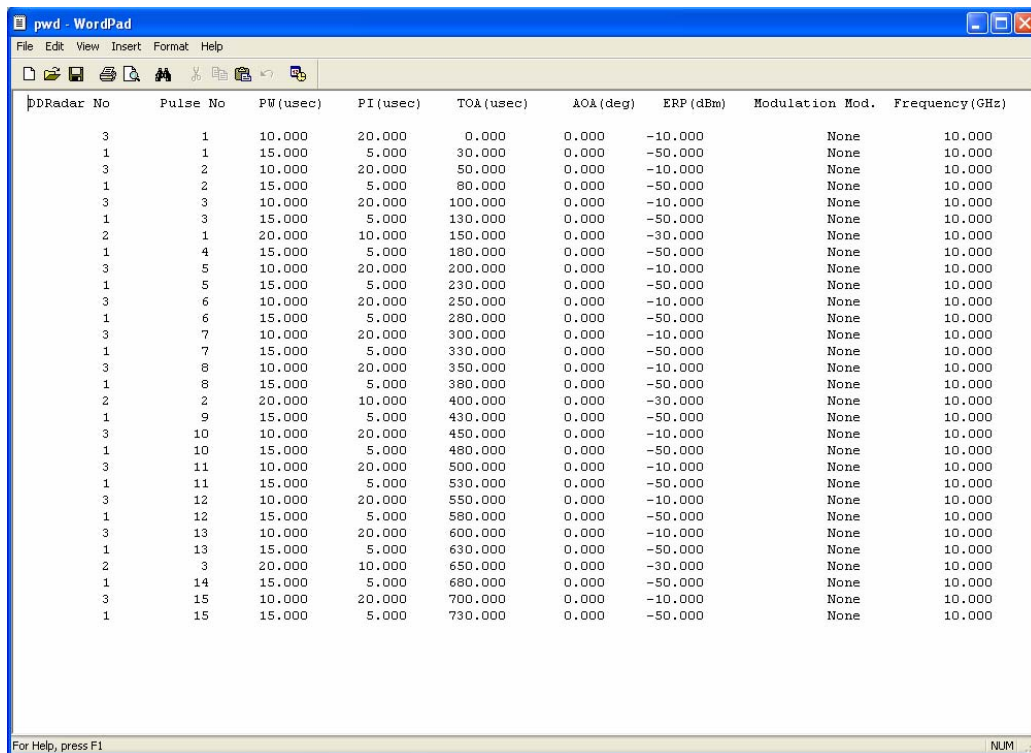
Figure 32

5.2. 'START' BUTTON :

This button is the main part of the program. When this button is clicked, the file chosen is read, the radars are separated, the data of each radar is identified, according the data the points are specified and finally with the help of two DLL these radars are graphed. Actually instead of the choice of the file all the work is done under this button's function. Since there are several actions under this button, it is much more reasonable to examine these events under different subheadings.

- 5.2.a. Reading the file:

First of all, the file should be opened. It is suitable to use fopen function like in the *Figure 33*. When the file is being read, a problem occurs; although most of the variables in the pwd file (one example is shown in *Figure 33*) are numbers and can be read as double numbers, the first several variables which show the names of the data are composed of characters and should be read as string. Also since in the modulation part of the radars is written 'none', these should be read as string, too.



pDRadar No	Pulse No	PW(μsec)	PI(μsec)	TOA(μsec)	AOA(deg)	ERP(dBm)	Modulation Mod.	Frequency(GHz)
3	1	10.000	20.000	0.000	0.000	-10.000	None	10.000
1	1	15.000	5.000	30.000	0.000	-50.000	None	10.000
3	2	10.000	20.000	50.000	0.000	-10.000	None	10.000
1	2	15.000	5.000	80.000	0.000	-50.000	None	10.000
3	3	10.000	20.000	100.000	0.000	-10.000	None	10.000
1	3	15.000	5.000	130.000	0.000	-50.000	None	10.000
2	1	20.000	10.000	150.000	0.000	-30.000	None	10.000
1	4	15.000	5.000	180.000	0.000	-50.000	None	10.000
3	5	10.000	20.000	200.000	0.000	-10.000	None	10.000
1	5	15.000	5.000	230.000	0.000	-50.000	None	10.000
3	6	10.000	20.000	250.000	0.000	-10.000	None	10.000
1	6	15.000	5.000	280.000	0.000	-50.000	None	10.000
3	7	10.000	20.000	300.000	0.000	-10.000	None	10.000
1	7	15.000	5.000	330.000	0.000	-50.000	None	10.000
3	8	10.000	20.000	350.000	0.000	-10.000	None	10.000
1	8	15.000	5.000	380.000	0.000	-50.000	None	10.000
2	2	20.000	10.000	400.000	0.000	-30.000	None	10.000
1	9	15.000	5.000	430.000	0.000	-50.000	None	10.000
3	10	10.000	20.000	450.000	0.000	-10.000	None	10.000
1	10	15.000	5.000	480.000	0.000	-50.000	None	10.000
3	11	10.000	20.000	500.000	0.000	-10.000	None	10.000
1	11	15.000	5.000	530.000	0.000	-50.000	None	10.000
3	12	10.000	20.000	550.000	0.000	-10.000	None	10.000
1	12	15.000	5.000	580.000	0.000	-50.000	None	10.000
3	13	10.000	20.000	600.000	0.000	-10.000	None	10.000
1	13	15.000	5.000	630.000	0.000	-50.000	None	10.000
2	3	20.000	10.000	650.000	0.000	-30.000	None	10.000
1	14	15.000	5.000	680.000	0.000	-50.000	None	10.000
3	15	10.000	20.000	700.000	0.000	-10.000	None	10.000
1	15	15.000	5.000	730.000	0.000	-50.000	None	10.000

Figure 33

For the reason stated in the previous page, first the first few variables are read as string in an array called 'array', second since there are six number until 'none' these are read as double and set to a different array called 'ARRAY'. Then after the word 'none' is read as string and sent to 'array', 10.000 which corresponds the unchanged frequency of the radar is set to ARRAY. As can be understood from the code in *Figure 34*, until the end of file this process will continue; first six variable are set to ARRAY, then one to array and one to ARRAY, later the next row's first six variable are set to ARRAY.....Finally an array of one dimension, which is called ARRAY, has been produced. It is good to remind that this array has the variables corresponding to 'radar no.'s at 'ARRAY[0], ARRAY[8], ARRAY[16]', or in other words at the multiples of 8.

```
//open the file
FILE *Dosya;
Dosya=fopen (m_sResults,"r");

//read the file
double okunan, ARRAY[2000];
char OkuString[4], array[100];
int end,k=0;

//note that it will first read the names as a string, then it will read the
values as double numbers except 'none's.
//it will also read 'none's as a string.
//then it will collect only the double numbers in an array named ARRAY

for (int i=0;i<=11;i++)
    fscanf (Dosya,"%s",array);

int l=0;

while( l<=6)
{
    end=fscanf( Dosya,"%lf",&okunan);
    ARRAY[k]=okunan;
    k++;
    l++;
}
```

Figure 34

```

end=fscanf( Dosya,"%s", OkuString);
end=fscanf(Dosya,"%lf",&okunan);
ARRAY[k]=okunan;
k++;

l=0;
while (end!=EOF)
{
    while( l<=6)
    {
        end=fscanf( Dosya,"%lf ",&okunan);
        ARRAY[k]=okunan;
        k++;
        l++;
    }

    l=0;

    end=fscanf( Dosya,"%s ",OkuString);
    end=fscanf(Dosya,"%lf",&okunan);
    ARRAY[k]=okunan;
    k++;
}

```

Figure 34 – continued

****** In the function the variable ‘k’ is used to calculate the size of the ARRAY.

- **5.2.b. Separating the radars’ data :**

To separate the radars, their radar numbers should be compared; for example if the radar number is 1 the next eight variable including itself will be collected in an array called ‘radar1’. To do this first the variables of the ARRAY which are the multiples of 8 should be compared with 1,2,3,4 and 5. However since the variables are read and set to the array as double numbers, it will not so easy to look whether the radar number is 1,2..... or 5. Initially the variable which corresponds to ‘radar no’ should be called and set to different variable as a string. On the other hand the double version of the number that will be compared should also be set to another variable (again as a string). After that it should be examined whether these are the same or not with the ‘compare’ function. For example

as shown in *Figure 35* to separate the first radar's data from others, at the beginning the double number which is the variable having the information of 'radar no' is set to a Cstring called 'x'. At the same time the double number 1.000 is set to another Cstring 'z'. Then, if these are the same this variable and the next seven variables are copied from 'ARRAY' to 'radar1' array. This procedure is same for all the radars. At the end all radars' data will have been separated. For example radar2 array only includes the data related with the second radar, that is, its first, ninth, seventeenth,..... (that are the variables that correspond to radar numbers {radar2[0],radar2[8],...}) variables are all 2.000.

```
//comparing the radar no.s separate the radar statistics
char birinci1[50],birinci2[50],birinci3[50],birinci4[50],birinci5[50];
char ikinci1[50],ikinci2[50],ikinci3[50],ikinci4[50],ikinci5[50];
double radar1[1000],radarsayisi=0.00;
double radar2[1000],radar3[1000],radar4[1000],radar5[1000];
int t,z,y,v=0,q=0,w=0,s=0,h=0;
for(z=0;;z++)
{
    y=z*8;
    if(y<k-8)
    {
        radarsayisi++;
        _gcvt(ARRAY[y],5,birinci1);
        _gcvt(1.000,5,ikinci1);
        CString x(birinci1);
        CString z(ikinci1);

        _gcvt(ARRAY[y],5,birinci2);
        _gcvt(2.000,5,ikinci2);
        CString L(birinci2);
        CString N(ikinci2);

        _gcvt(ARRAY[y],5,birinci3);
        _gcvt(3.000,5,ikinci3);
        CString G(birinci3);
        CString S(ikinci3);

        _gcvt(ARRAY[y],5,birinci4);
        _gcvt(4.000,5,ikinci4);
        CString R(birinci4);
        CString T(ikinci4);

        _gcvt(ARRAY[y],5,birinci5);
        _gcvt(5.000,5,ikinci5);
        CString Y(birinci5);
        CString U(ikinci5);
    }
}
```

```

        if(0==x.Compare(z))
        {
            for (t=y;t<y+8;t++)
            {
                radar1[v]=ARRAY[t];
                v++;
            }
        }
        else if(0==L.Compare(N))
        {
            for (t=y;t<y+8;t++)
            {
                radar2[q]=ARRAY[t];
                q++;
            }
        }
        else if(0==G.Compare(S))
        {
            for (t=y;t<y+8;t++)
            {
                radar3[w]=ARRAY[t];
                w++;
            }
        }
        else if(0==R.Compare(T))
        {
            for (t=y;t<y+8;t++)
            {
                radar4[s]=ARRAY[t];
                s++;
            }
        }
        else if(0==Y.Compare(U))
        {
            for (t=y;t<y+8;t++)
            {
                radar5[h]=ARRAY[t];
                h++;
            }
        }
    }
    else
        break;
}

```

Figure 35

- **5.2.c. Graphing the radars :**

Actually when it comes graphing, a function called 'cvLine' is a helpful. This function draws straight lines between two points which should be stated as 'point1' and 'point2'. That is, if the first and last points' coordinates are specified, with the help of 'cvLine' function these points can be combined with a straight line. Also this function gives the programmer the opportunity of changing the line's thickness and color.

Since the drawing will be done on a static function, first the static text on the lower part of the dialog window is created and named as 'Resim'. Then the boundaries of this text should be specified with the formula written in the second row of *Figure 36*. In this program the size of x-axis is specified as 1000 while the size of y-axis is 100.

When the code is continued to examine, it is seen that one radars variables are set to a common array called 'arrayy', otherwise the code will be too longer. At the same time since it is desirable that all the radars' colors are different, the variable 'r' which will the color number is set to different values.

Actually in this part the program works like that;

First it starts with the first radar, takes the 'radar1' array, puts its all variables to 'arrayy' and specifies its color as the color corresponding to 255. Also the size of the array is set to the variable 'p'. (actually this is useful to finish the graph when the radar is stopped.) Then 'arrayy' is drawn with 'cvLine' function. When the first radar's drawing is completed, the number will be increased to 2 in for loop. That is now the variables of 'radar2' will be set to 'arrayy' and the second radar is drawn. This process will stop when the fifth radar's drawing is completed.

**** Note that in *Figure 36* the for function has not yet finished, it will continue in *Figure 37*.**


```

//drawing
IplImage* Resim = cvCreateImage(cvSize(1000,100), IPL_DEPTH_8U, 4);

//for the different radars set the values of color,length and array values differently
double A,B,C,D,E=80.00;//h
int p,r,j;
double arrayy[1000];

for(j=1;j<6;j++)
{
    if(j==1)
    {
        p=v;
        r=255;
        for(i=0;i<1000;i++)
            arrayy[i]=radar1[i];

    }
    else if(j==2)
    {
        p=q;
        r=355;
        for(i=0;i<1000;i++)
            arrayy[i]=radar2[i];
    }
    else if(j==3)
    {
        p=w;
        r=455;
        for(i=0;i<1000;i++)
            arrayy[i]=radar3[i];
    }
    else if(j==4)
    {
        p=s;
        r=555;
        for(i=0;i<1000;i++)
            arrayy[i]=radar4[i];
    }
    else
    {
        p=h;
        r=155;
        for(i=0;i<1000;i++)
            arrayy[i]=radar5[i];
    }
}

```

Figure 36

Drawing the ‘arrayy’: In this part of the code, first the radar is examined whether its first pulse starts at 0 or later. If the time of arrival of the first pulse is not zero, a horizontal straight line should be drawn at -80dBm (it is the magnitude of the ‘OFF’ state) until the first pulse starts. Then first the vertical line, second the horizontal line at the magnitude of the pulse, third again the vertical line which completes the pulse and finally a horizontal line at -80dBm which represents the waiting time until the next pulse are plotted with the help of ‘cvLine’ function and the necessary variables of ‘arrayy’. Figure 37 shows the code that corresponds the events mentioned above.

```
//if the first pulse doesnt arrive at 0, draw the the line until the pulse
    A=00.00;
    C=arrayy[4];
    B=E;
    D=E;

    CvPoint pt1;
    CvPoint pt2;
    pt1.x=A; pt1.y=B;
    pt2.x=C; pt2.y=D;

    if (pt2.x!=0)
    {
        cvLine( Resim, pt1, pt2, CV_RGB(r,255,0), 1, 8);
    }

    //draw the pulses using the first point and the last point of each distinct line
    for(i=4;i<=p-4;i+=8)
    {
        A=arrayy[i];
        C=arrayy[i];
        B=E;
        D=-arrayy[6];//h

        CvPoint pt1;
        CvPoint pt2;
        pt1.x=A; pt1.y=B;
        pt2.x=C; pt2.y=D;
        cvLine( Resim, pt1, pt2, CV_RGB(r,255,0), 1, 8);

        //cvShowImage("RESIM", Resim);
```

Figure 37

```

        A=arrayy[i];
        C=arrayy[i]+arrayy[i-2];
        B=-arrayy[6];//h
        D=-arrayy[6];//h

    pt1.x=A; pt1.y=B;
        pt2.x=C; pt2.y=D;
    cvLine( Resim, pt1, pt2, CV_RGB(r,255,0), 1, 8);

        //cvShowImage("RESIM", Resim);

        A=arrayy[i]+arrayy[i-2];
        C=arrayy[i]+arrayy[i-2];
        B=-arrayy[6];//h
        D=E;

    pt1.x=A; pt1.y=B;
        pt2.x=C; pt2.y=D;
    cvLine( Resim, pt1, pt2, CV_RGB(r,255,0), 1, 8);

        //cvShowImage("RESIM", Resim);

    if(i!=p-4)
    {
        A=arrayy[i]+arrayy[i-2];
        C=arrayy[i+8];
        B=E;
        D=E;

    pt1.x=A; pt1.y=B;
        pt2.x=C; pt2.y=D;
    cvLine( Resim, pt1, pt2, CV_RGB(r,255,0), 1, 8);
    }
    }
    }

    cvShowImage("RESIM", Resim);

    fclose(Dosya);

    cvReleaseImage(&Resim);
}

```

Figure 37 – continued

In addition, graphing some small lines showing the important values in the screen will be very beneficial for the user. It will provide the user to understand the graph. For this reason using the same function 'cvLine', straight lines in the color of grey are drawn with the code in **Figure 38**. In the y-axis, the lines with a size of 2 are drawn. On the other hand in the x-axis except the lines at the points 100,200,300,... the size of the lines are 3(=100-97). At that points the lines are at the size of 10(=100-90).

```
//draw the lines for coordinates
int m=1;
for(i=0;i<=100;i+=10)
{
    CvPoint pt1;
    CvPoint pt2;
    pt1.x=0; pt1.y=i;
    pt2.x=2; pt2.y=i;

    cvLine( Resim, pt1, pt2, CV_RGB(255,255,255), 1, 8);
}

for(i=0;i<=1000;i+=10)
{
    CvPoint pt1;
    CvPoint pt2;
    pt1.x=i; pt1.y=100;
    pt2.x=i;

    //for 100 and its multiples draw longer lines
    if(i==100*m)
    {
        pt2.y=90;
        m++;
    }
    else
        pt2.y=97;
}
```

Figure 38

5.3. 'EXIT' BUTTON :

In the dialog window an 'exit' function should be placed to close the program. For this purpose a button with an ID of EXIT should be activated. That is a function of it should be

created and with a proper code the task of closing the window should be loaded to this button. Indeed, in VISUAL C++ this can be done with a simple OnOK() function like seen in *Figure 39*.

```
void CNewexerciseDlg::OnExit()
{
    // TODO: Add your control notification handler code here
    OnOK();
}
```

Figure 39

Actually this is the end of the program designed to graph the radars created in the software RADSIM. How the user runs the program can be repeated like that;

When the file is chosen as the pwd file and then when the start button is clicked, the radars will have been represented graphically like in the *Figure 40*.

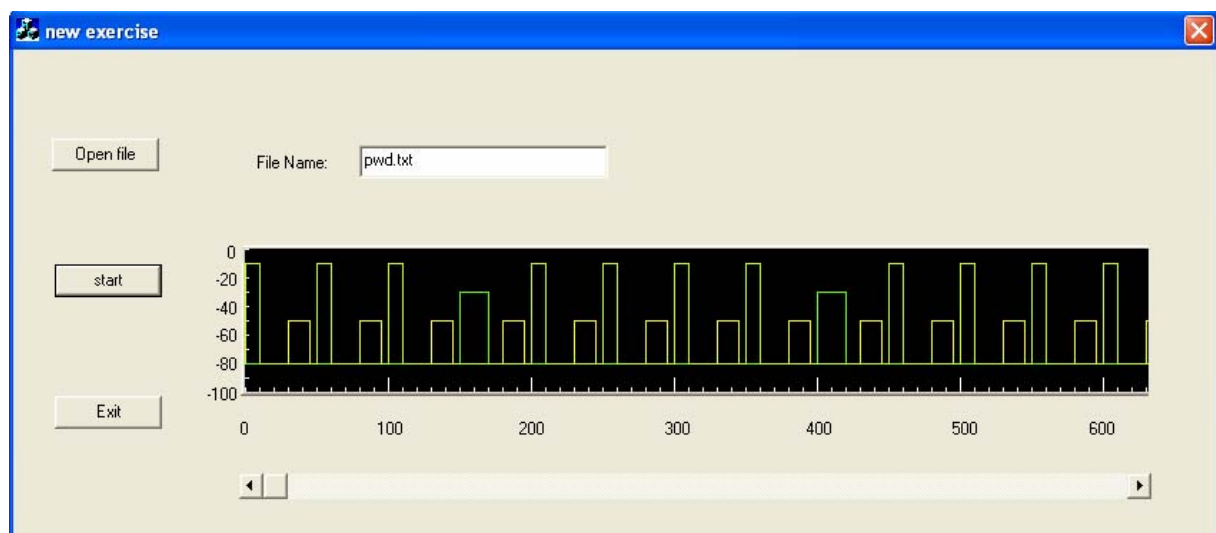


Figure 40

Actually if the same file is represented graphically with VEE as in *Figure 41_a* & *Figure 41_b*, it is seen that VEE has much more facilities for the user. The user of the program can see the radars separately; he can zoom in or out the screen; he can see the points used to draw the graph.... In fact the graph constructed with Visual C++ is much more limited than VEE.

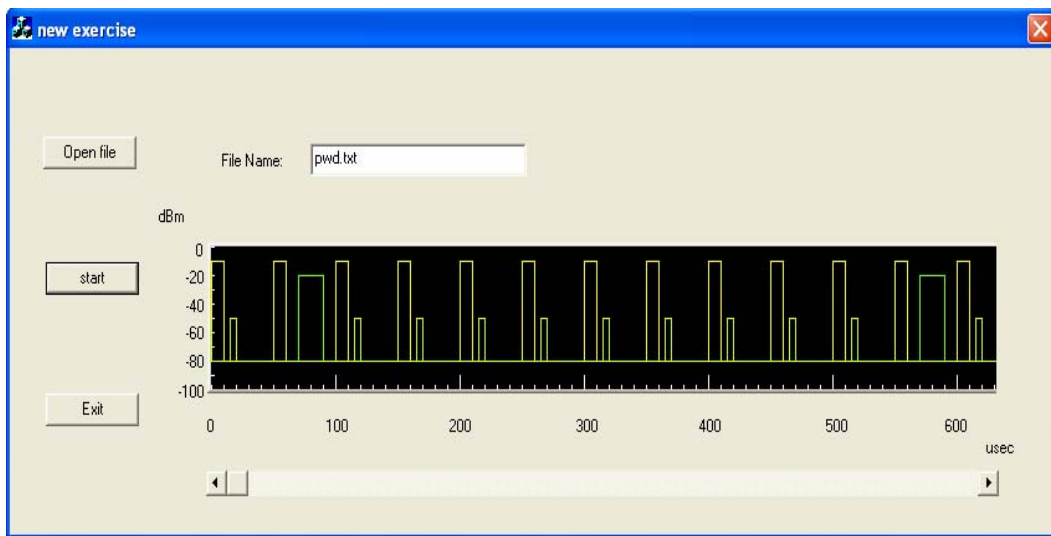


Figure 41_a

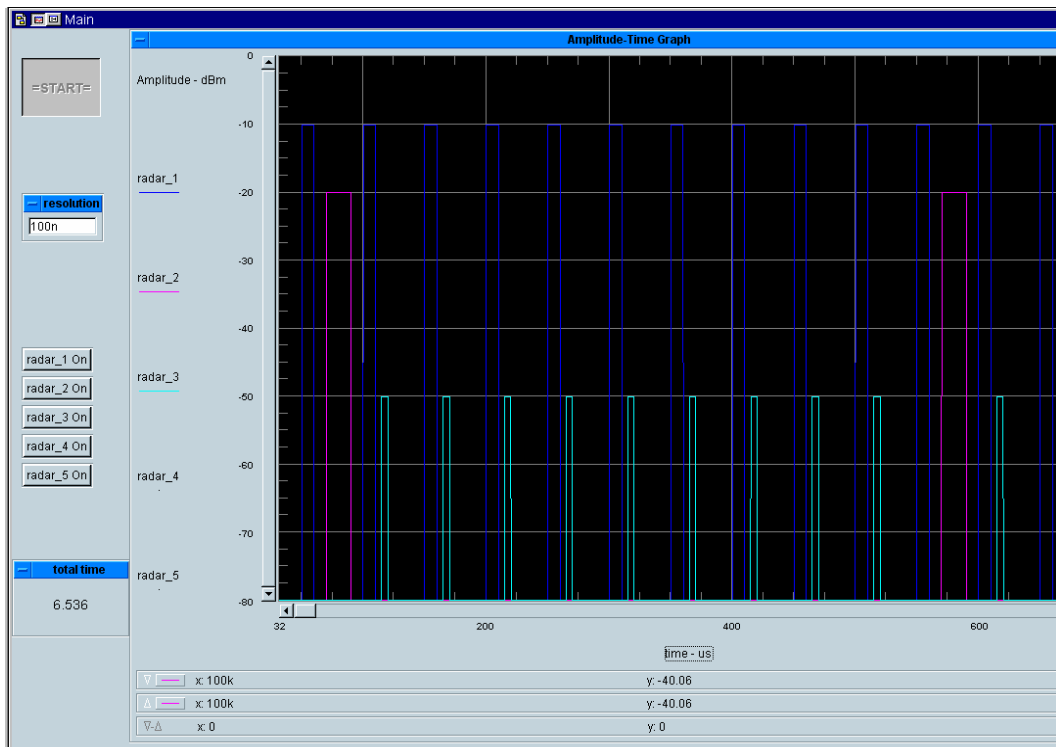


Figure 41_b

6. CONCLUSION :

When I learned that I would done my SP in ASELSAN, I was appreciated to be a part of ASELSAN although it was true for a temporary time. Because I have heard that ASELSAN is one of the most important companies for the Electrical and Electronic Engineers in Turkey. In fact when I was doing my SP, I saw that it was true. It has really high standarts of technology. However in my opinion the reason why it is one of the big companies in Turkey is the discipline in the company. Indeed all the workers pay really attention to their duty and all the works are done professionally. In addition, during my SP I had a chance of observing many things about business life. First of all, I have understood that a big effort is needed to be a fine engineer. You should always expand your knowledge, in other words you should improve yourself.

On the other hand, if I have talked about the benefits that I got, first of all while doing my work I had learned two programming languages. Since I have worked with radars which I did not have so many information, it was good to learn somethings about radars, their characteristics and their functions. On the other hand, may be the programmes I have designed will be used in some works in future. Indeed it is good to know someone will benefit from your works. May be the best sentence that will explain the benefits of SP is that in the summer practice you understand that producing new things is a very exciting and interesting task.

Actually I can really recommend this location to all the students for business life. Because this company is one step further from most companies in Turkey with its facilities, technological opportunities,etc. And I believe it will continue to be one of the important companies of Turkey.

7. REFERENCES :

- <http://wikipedia.org/wiki/Radar>
- [http:// www.agilent.com/find/vee](http://www.agilent.com/find/vee)
- “Agilent VEE Pro User’s Guide”, Agilent Technologies, 2004
- “Programming with Visual C++”, David J. Kruglinski, 1998
- “Sams Teach Yourself Visual C++ 6 in 21 days”, Macmillan Computer Publishing