MONITORING AND ADVISING SYSTEM FOR IODINE STATUS

MONITORING AND ADVISING SYSTEM FOR IODINE STATUS

By

LOBAT TAYEBI, B.S.

A Thesis Submitted to the School of Graduate Studies in Partial Fulfillment of the Requirements for the Degree of M.A.Sc.

McMaster University

M.A.Sc. (2006) **McMaster University**

(Engineering Physics) **Hamilton, Ontario**

TITLE: Monitoring and Advising System for Iodine Status

AUTHOR: Lobat Tayebi, B.S. (Sharif University of Technology**)**

SUPERVISOR: Professor W.J. Garland

NUMBER OF PAGES: XIII, 144

Abstract

Iodine-125 is produced from Xe-124 at McMaster Nuclear Reactor (MNR) for medical applications. Excessive amounts of this isotope in the air of the reactor site can be hazardous especially for the staff members who are continuously working at the reactor building. Therefore there is a critical need to monitor the concentration of iodine released to the air. In the previous iodine monitoring system available at MNR, there were several technical and practical problems. Consequently a new monitoring system was designed to overcome the previous problems and to provide with further facilities for the staff. The system is developed in the LabVIEW[1] platform and is easily maintainable. The external data acquisition board LabJack is used to transfer the data to the PC. Briefly, by using radiation monitoring techniques and statistical methods, a computerized embedded system is designed to keep the MNR a safe place for operators and visitors. This thesis examines in detail the design and implementation of the new monitoring system, MASIS.

[1] National Instruments LabVIEW™ (Laboratory Virtual Instrument Engineering Workbench) is a software tool for designing test, measurement, and control systems.

This thesis is dedicated to my parents, Mohammad and Shahnaz, and my loving husband, Daryoosh.

Acknowledgments

My grateful thanks go to my supervisor, Dr. W.J. Garland for his support and understanding in doing this work, which have permitted me to achieve not only my Master‘s degree, but inner satisfaction and peace.

Great appreciate is also offered to Heinz Schlichting from Health Physics department who was fully involved from the beginning to the end of the project.

I also wish to express my gratitude to Dave Tucker for his constant feedback and fruitful discussions on MASIS project.

Appreciation is also directed to Alice Pidruczny, Steve Staniek, Norbert Boehling, Barry Diacon, Glen Leinweber for their helps in different parts of the project.

Also, special thank goes to all staff of the McMaster Nuclear Reactor, whom it was a great pleasure to work with.

Last but not least, thanks to my husband, Daryoosh, and my parents, Mohammad and Shahnaz, for their continuing encouragement in these two years.

Monitoring and Advising System for Iodine Status

Table of contents

Chapter **Page number**

List of Figures

Figure **Page number**

Chapter 1

Introduction

Iodine-125 that is mainly used for medical purposes is one of the important products of McMaster Nuclear Reactor (MNR). Doses of this radioactive gas that are higher than the maximum allowable amount can be very harmful especially for the thyroid organ. For this reason it is necessary for the reactor to monitor iodine-125 contained in the air. This monitoring is critical for the workers who work continuously in the reactor to assure a safe work place.

The existing IMAS (Iodine Monitoring and Advising System) has been employed for monitoring the amount of iodine in MNR since 2000. However, it was later found that IMAS had several problems that needed to be resolved. In addition, the operators wished to have some extra features that IMAS was not able to provide. Therefore a new system was planned to overcome these problems and provide the features that the operators required. This new system, MASIS (Monitoring and Advising System for Iodine Status), is the subject of this MSc thesis. In this introduction, a brief description of MASIS is provided and the subject of each chapter of the thesis is presented, giving an overview for this manuscript.

In the following chapter, the concepts, calculations and motivations of MASIS are described in detail. This also includes the problems of the old monitoring system that provided the motivation for a new system. Some of the major problems relate to IMAS upgradeability, problems with the alarm and annunciation system, system crashing, and poor error handling.

Also in that chapter we will see the I-125 is produced from Xe-124 through this chain reaction:

$$^{124}Xe \xrightarrow{(n,\gamma)} {}^{125}Xe \xrightarrow[17h]{(\beta,\gamma)} {}^{125}I$$

Estimation of Iodine Concentration is the most important concept in MASIS that is explained in this chapter. Equations are derived for the related parameters. "Slope of count-rate in time" is calculated by the method of least-squares solution.

Also the equation for estimation of error in Iodine Concentration is derived in Chapter 2: "Concepts and Motivations for MASIS"

The next chapter, Chapter 3, is about the hardware of the system. The hardware mainly contains an air pump, a charcoal filter, an NaI detector, a data acquisition board called LabJack U12, a PC, and a Signal Tower. The air pump passes the air of the site to the charcoal filter. The filter traps the iodine-125 contained in the air and the NaI detector detects the new iodine arrived. The 32 bit counter in the LabJack U12 counts the signals from the NaI detector and sends the result to the LabVIEW program for analysis. Signal Tower is the main part of the alarm and annunciation system in the MASIS. It includes three lights (green, amber, red) and a buzzer to inform the reactor crew of the different levels of Iodine Concentration in the air. LabJack U12 also passes the signals from the LabVIEW program, through some digital I/O ports, to another switchboard that controls the signal tower.

Adding to these main parts of the hardware, a 4-point solid-state-relay called TriacOut4 is used as a switchboard between LabJack U12 and the Signal Tower. Also a transformer in the system converts the line voltage to the voltage suitable for the Signal Tower (i.e. 24 V AC). All of these parts are explained in detail in the Chapter 3.

Chapter 4 is "Graphical Representation of the Process in the Main Program: Flowchart". In this chapter, the main procedures in MASIS are divided into four parts:

1-Counting the Signals

2-Considering Two Different Modes: Normal Mode and Filter Change Mode

3-Calculating Iodine Concentration and Demonstrating the Results

4- Alarm and Annunciation Activity

The flowchart of each part is presented and explained in this chapter. "Counting the Signals" is mainly about communicating with the counter of LabJack U12 through the CNT port, counting process, and reporting the count-rate.

Section of "Considering Two Different Modes: Normal Mode and Filter Change Mode" described two available modes in the program. When the amount of iodine in the filter increases, the sensitivity of the system is decreased and the detector can not record the new iodine arrived into the system. This is the time that the error of Iodine Concentration is very high, so the filter must be changed. Usually the charcoal filter is changed once a week. It normally takes ten to fifteen minutes to change the filter. At this time, the program is set to stay in an infinite loop. Therefore, it can not calculate the Iodine Concentration because obviously the count-rate is not meaningful at this time. The period of changing the charcoal filter is called "Filter Change Mode". After changing the filter, MASIS needs an initializing time, usually one hour, to collect the data needed for accurate estimation of Iodine Concentration. Also after starting the program, the code needs this initializing time to gather the prerequisite count-rate data. When the program is neither in the initializing period nor Filter Change Mode, it is in the Normal Mode.

Next part of the main flowchart, "Calculating Iodine Concentration and Demonstrating the Results" is the heart of the program. After counting the signals and considering two modes, the program calculates the Iodine Concentration. If the program is not in the initializing period, the result is recorded and displayed through the corresponding graphs and indicators in the main screen. The detailed description of these processes is discussed in this section. Also the error of Iodine Concentration at each time step is estimated in this section.

"Alarm and Annunciation Activity" is the last section of Chapter 4. For alarm and annunciation, MASIS has a Signal Tower with three lights and a buzzer that is mounted on the wall above the system. Also, three flashing lights on the main screen mimic the lights of the Signal Tower. Green lights indicate the normal condition. Also, when this light is "On", there is no problem with the system. In case of any error in the system, the green light turns off. During the initializing period and Filter Change Mode the green light is "Off".

The amber light turns on when iodine release is occurring. It indicates warning condition.

The red light and also the buzzer are activated when the level of iodine approaches the administrative release limit. The flashing red light is for the alarm condition.

One of the features of the alarm and annunciation system in MASIS is the ability to send out email messages in alarm conditions. This is done automatically when the red light turns on. A data file is also attached to the email that shows the detail description of the situation at the alarm condition. This feature is also explained in this section of the flowchart chapter.

"Code Description" is the next and probably the most important chapter of the thesis. This chapter is dedicated to show the complete MASIS code and discuss the details. The code of MASIS is written in LabVIEW 7.1. This chapter, after a brief explanation of LabVIEW, goes through the four main parts that are described in the flowchart chapter to show the related code and explain the functions and Sub-VIs that were used in these parts. Also the next three sections of this chapter clarify parts of the code related to the appearance and execution of the program. These three sections are: "Creating files, writing into and reading from them", "Plotting the Graphs" and "Making and Controlling the Password Protected Area".

The last chapter of the thesis is a brief conclusion and some suggestions for the future works.

Chapter 2

Concepts and Motivations for MASIS

Introduction

Since there were several problems with IMAS, the old system of iodine monitoring in the McMaster Nuclear Reactor (MNR), the reactor management team decided to have a new system that can accommodate these problems and add some desired new features. In this thesis, a detailed description of the new system called MASIS (Monitoring and Advising System of Iodine Status) will be presented. In this chapter, a brief description of the old system and its problems that was the motivation to design a new system is explained. Then the basic concepts and mathematical calculations related to the new system, such as estimation of Iodine Concentration and error calculations, are discussed. These concepts and analysis are used extensively in the following chapters of the thesis.

2.1 Old System Deficiencies

The old monitoring system in MNR known as IMAS (Iodine Monitoring and Advising System) is working under the Windows NT environment and is written in the DELPHI language. This system is shown in Figure 2-1.

A Keithley Metrabyte CTM-05/A is used as a counter-timer board that transfers the signal into the PC. There have been several technical and practical problems with IMAS. These problems can be listed as follows:

-It is not easily upgraded to the new generation computer equipment.

-The software is developed on the DELPHI platform that not many people are familiar with. Lack of access to the full source code makes it impossible to upgrade or migrate.

-With this system we just have the current data. If data analysis of a specific past period of time is needed, the raw data cannot be accessed.

-There is insufficient error trapping in the numerical algorithms. System crashes can occur.

-The system is tied to special hardware. For example, it requires a specific interface device to function (like the Keithley card). Newer PC's do not have facilities (i.e. expansion slots) to accommodate such a device.

- Alarm conditions and the way they are displayed can be confusing if the observer is not familiar with the IMAS system. There are too many (often, not easily understood) conditions that will trigger an alarm.

-There is no visual alarm indicator other than the monitor display. This does not allow anyone not in the immediate area of the IMAS station to be aware of an alarm condition.

-Audible alarm annunciations are too repetitive and cannot be cancelled (even for momentary relief). The annunciation will stop only after the alarm condition clears or the alarm parameters are manually adjusted.

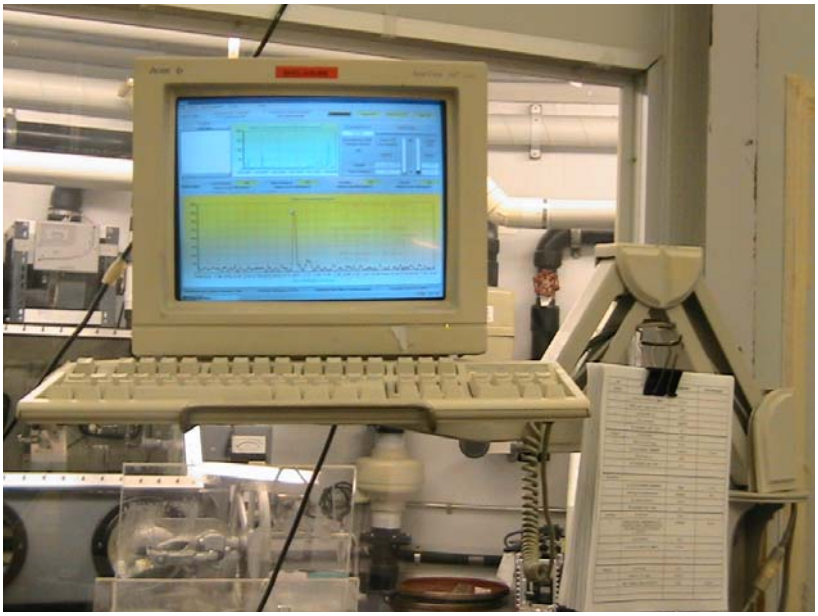


Figure 2-1: IMAS (Iodine Monitoring and Advising System)

2.2 Iodine-125

Among the isotopes produced in MNR, Iodine-125 is the most important one. This radioactive gas is a widely used material and demand for it is increasing. It is employed for diagnostic and therapeutic applications. Some of these applications are: in-vitro diagnostic kits (radioimmunoassay), as a source for bone densitometry devices, protein iodination and therapeutic seed implants used in the treatment of prostate cancer, and also as a guide for surgeons in the localization of cancerous nodes in radio-immunoguided surgery (RIGS)[1].

To produce I-125, Xe-124 gas captures a neutron to form Xe-125 which decays with a half- life of 17 hours to form I-125.

$$^{124}Xe \xrightarrow{(n.\gamma)} {}^{125}Xe \xrightarrow[17h]{(\beta,\gamma)} {}^{125}I$$

Although the radiation that is emitted by this isotope of iodine is of a low energy, it can be very dangerous for human's body. A very small amount can be very harmful for the thyroid organ. For the people who are working 40 hours in a week in the reactor, the maximum allowable airborne I-125 concentration is 80 Bq/m3.

MNR arranges tours for interested people especially students. So to have a protected place for operator, visitors and researchers, the reactor needs to have an accurate I-125 monitoring system.

2.3 Iodine Concentration

The concentration of iodine is the most important parameter in the MASIS. Iodine Concentration is the measure for the amount of airborne iodine in the reactor site. All the decisions for the alarm and annunciation activity and determining the present situation are based on this parameter. Therefore, it is required to convert the count-rate of the detected I-125 to the Iodine Concentration in Bq/m^3.

The half-life of iodine is about sixty days. There is a charcoal filter in the system that captures the iodine in the air pumped by the air pump. The filter is being changed almost every seven days. To show that it is reasonable to ignore the amount of iodine that is lost by decay in the filter, assume the amount of iodine trapped in the filter has a constant rate of R. If λ is the iodine decay constant, we have:

$$\frac{dI}{dt} = R - \lambda I \qquad \textbf{Equation 1}$$

The solution of this differential equation is:

$$I = \frac{R}{\lambda}(1 - e^{-\lambda t}) \qquad \textbf{Equation 2}$$

The amount of iodine in the filter without decay is defined I' ($I' =$ Rt). We compare the ratio:

$$\frac{I}{I'} = \frac{(1 - e^{-\lambda t})}{\lambda t} \qquad \textbf{Equation 3}$$

As λ is $\frac{\ln 2}{60 days}$ and t is almost one week, λt is small. Therefore, with a good approximation $\frac{I}{I'} = 0.96 \cong 1$ and we can ignore decay in our calculation.

As a result, the Iodine Concentration is directly proportional to the rate of iodine trapped in the filter or the slope of the count-rate that is acquired from the NaI detector. The NaI detector is counting the amount of iodine that comes through the air pump and is trapped by the charcoal filter. For this reason, the rate of change of iodine in the filter is proportional to the Detector Efficiency and Pumping Flow Rate too[2]. Hence, we can write the following relation:

$$\text{Rate of change of Iodine counted in the filter} \propto \text{Iodine concentration} \times \text{Detector Efficiency} \times \text{Pumping Flow Rate}$$

Equation 4

The filter efficiency is almost one. The rate of iodine counted in the filter is proportional to the slope of the count-rate in time. Consequently, we will have this relation:

$$Slope\ of\ \ count\ rate\ in\ time = Iodine\ concentration \times Detector\ Efficiency \times Pumping\ Flow\ Rate$$

Equation 5

or:

$$Iodine\ concentration = \frac{Slope\ of\ \ count - rate\ in\ time}{Detector\ Efficiency \times Pumping\ \ Flow\ Rate}$$

Equation 6

Here the unit of Iodine Concentration is Bq/m^3 (1 Bq= 1 disintegration /second). The unit for the slope is Bq/s, the Pumping Flow Rate is stated in m^3/s and detector efficiency is in percent.

But as we usually have Pumping Flow Rate in L/min, we can convert Equation 6 to this equation:

$$Iodine\ concentration = 6\times10^{6}\frac{Slope\ of\ \ count\ rate\ in\ time}{Detector\ \ Efficiency\times Pumping\ \ Flow\ Rate} \qquad \textbf{Equation 7}$$

Note: 1 m^3=10^3 Liter, 1min=60s and %=10^{-2}.

To find the "slope of count-rate in time" the least-squares solution is employed. To have a brief description of this method, assume that we have some experimental data in a period of time shown in Figure 2-2-2.

We assume a model: $Y(t)=mt+b$. When $0\le t\le T$, N samples is obtained and $y(n)$, where $n=0,1,...,N-1$, is the real data corresponding to $Y(t)$, where $t=n\Delta$ and $\Delta=1$ time unit.

So in the discrete-time signal model: $y(n)=mn+b$.

If the least-squares solution is used, we have the following estimation for slope and intercept[3]:

$$m=-\frac{6}{N(N+1)}\sum_{n=0}^{N-1}y(n)+\frac{12}{N(N^2-1)}\sum_{n=0}^{N-1}ny(n) \qquad \textbf{Equation 8}$$

$$b=\frac{2(2N-1)}{N(N+1)}\sum_{n=0}^{N-1}y(n)-\frac{6}{N(N+1)}\sum_{n=0}^{N-1}ny(n) \qquad \textbf{Equation 9}$$

We fit the experimental data shown in Figure 2-2-2 by $Y(t)=mt+b$ with the above m and b.

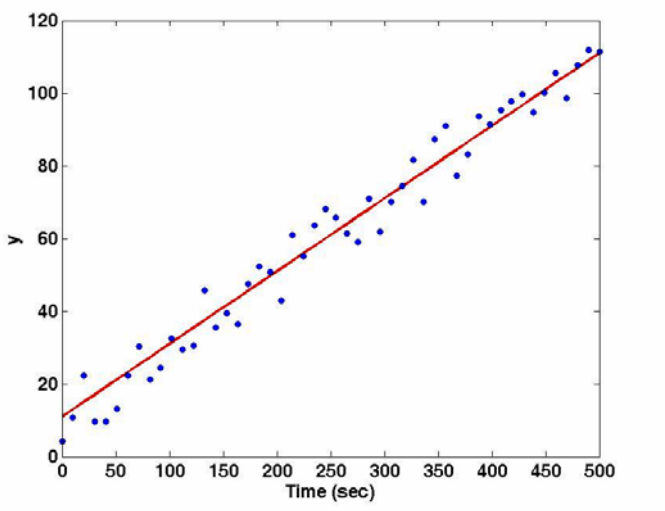


Figure 2-2-2: Typical experimental data in 500 seconds and the best-fitted line approximated by least-square solution.

Since in our program the count-rate has a form of a linear ramp, the assumption of $Y(t) = mt + b$ is completely reasonable and only calculating the slope (m) is needed. To find the more accurate result for Iodine Concentration, at each second the data of the last hour is considered.

We are approximating the count rate with a piece-wise linear function where the step is the sample length. If the sample length (for example one hour) is smaller than the change of count rate, the approximation is accurate. The sample length can be adjusted in MASIS. The operator can reduce the sample length to get a faster response. However, if the sample length is too small, the estimation of the slope is not accurate due to the increase of noise effect. More data can better cancel the noise effect. On the other hand, if the time scale of the change is small compared to the sample length, the Iodine Concentration is underestimated. In practice a sample length of one hour covers a typical airborne Iodine dynamics.

For the reason mentioned above, a sample length of one hour may underestimate rapid changes in the Iodine Concentration. However, it is possible to add a separate alarm condition to announce rapid changes in the Iodine Concentration. In the current system, the operators decided not to have such an alarm situation to avoid possible confusion.

However, since in addition to the Iodine Concentration, the actual count rate is also shown at each time step on the main screen, the operator can observe any rapid change of Iodine Concentration.

2.4 Error Estimation

While the air is pumped by the air pump, a charcoal filter traps the air-born iodine to be detected by the detector. When the amount of iodine increases in the filter, the accuracy of the system decreases because the detector is less sensitive to the new iodine when the filter is almost full. At this time the filter has to be changed. Thus, we need to know the error each time that the Iodine Concentration is calculated, especially when the filter is not new.

To find the error, assume "Y" is the estimation and "y" is the real count-rate. The mean square error between "Y" and "y" is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n-1}(Yi - y_i)^2 \qquad \textbf{Equation 10}$$

We can say that:

$$Error = \sqrt{\frac{1}{n}MSE} \qquad \textbf{Equation 11}$$

So simply:

$$y_i = Y_i \pm Error_i \qquad \textbf{Equation 12}$$

$$y_{i-1} = Y_{i\_1} \pm Error_{i-1} \qquad \textbf{Equation 13}$$

By subtracting Equation 12 and Equation 13, dividing both sides by Δt and multiplying by K, Equation 14 is extracted:

$$K\frac{y_i - y_{i-1}}{\Delta t} = K\frac{Y_i - Y_{i-1}}{\Delta t} \pm K\frac{Error_i - Error_{i-1}}{\Delta t} \qquad \textbf{Equation 14}$$

Assume:

$$K = \frac{6\times 10^6}{Detector\ Efficiency \times Pumping\ Flow\ Rate} \qquad \textbf{Equation 15}$$

Then from Equation 14:

$$Iodine\ concenration_{(real)} = Iodine\ concentration_{(estimated)} \pm K\frac{\Delta Error}{\Delta t} \qquad \textbf{Equation 16}$$

So:

$$percentage\ of\ the\ error = \pm 100\frac{Iodine\ concentration_{(real)} - Iodine\ concentration_{(estimated)}}{Iodine\ concentration_{(estimated)}} = \pm 100\ K\frac{\frac{\Delta Error}{\Delta t}}{Iodine\ concntration_{(estimated)}}$$

Equation 17

The percentage of the error is low (less than 5%) just after filter changing and then gradually increases .Usually after one week the error is not acceptable (about 25%) and the filter should be changed.

2.5 Testing and Verification

A complete industrial verification of the system is not in the scope of this thesis. However, different actions have been taken to verify the reliability of MASIS in the estimation of the airborne I-125 concentration and its response in the alarm condition.

The parameters that are used to estimate the Iodine Concentration are count rate, Detector Efficiency and Pumping Flow Rate. The values of the Detector Efficiency and

Pumping Flow Rate are input data to MASIS. These are constant numbers measured by the operator every week. Any error in these parameters is not considered as an error in MASIS and should be addressed in their own measurement process.

However, the input data for the calculation of Iodine Concentration in MASIS is based on the I-125 count-rate. Therefore, validating the count rate is perhaps the most important part of the system verification. The 32 bit counter in the LabJack U12 counts the signals coming from the NaI detector. The LabVIEW software with the use of LabJack functions is able to communicate with this counter and quantify the count rate.

In order to verify if the measured count is equal to the real count rate, the NaI detector has been simulated by an external signal generator. A square-wave periodic signal with a known frequency from the signal generator was input to the LabJack counter terminal. The count-rate measured by MASIS system was equal to the input frequency, which confirmed the accuracy of the count-rate calculations.

MASIS has been installed in the reactor since February 2006. Until now it has been working properly in all different situations. In all cases the system shows the expected response. For example, when the operators are working on producing I-125, the system shows the increase on Iodine Concentration properly and on time. Other parts of the system (such as the lights and the buzzer, manual log entry, etc.) work appropriately as well. The last several months of testing the system in the actual environment has shown the acceptable reliability of the system.

Chapter 3

System Hardware

Introduction

The main parts of the hardware of the MASIS system are an air pump, a charcoal filter as an iodine trap, an NaI detector, LabJack U12, a PC and the Signal Tower. The System also includes a TriacOut4 board as a switch between LabJack U12 and the Signal Tower and a transformer that converts the 120V AC from the line to 24V AC which is suitable for the Signal Tower.

Figure 3-3-1 shows the layout of the system hardware. The air pump pumps the air into the charcoal filter and the filter captures the I-125 contained in the air. The charcoal filter has the capability of trapping 99% of the present iodine 125. When the new iodine arrives, the Nal detector shows a rise in the filter activity. The signal peaks from the detector are counted by the LabJack U12 and reported at the PC. The LabJack U12 is a data acquisition board that not only is used as a counter but also provides output signals to control the Signal Tower. The Signal Tower with three lights and a buzzer responds to different levels of iodine concentration at different situations (normal, warning and alarm). These levels are calculated and decided by the PC.

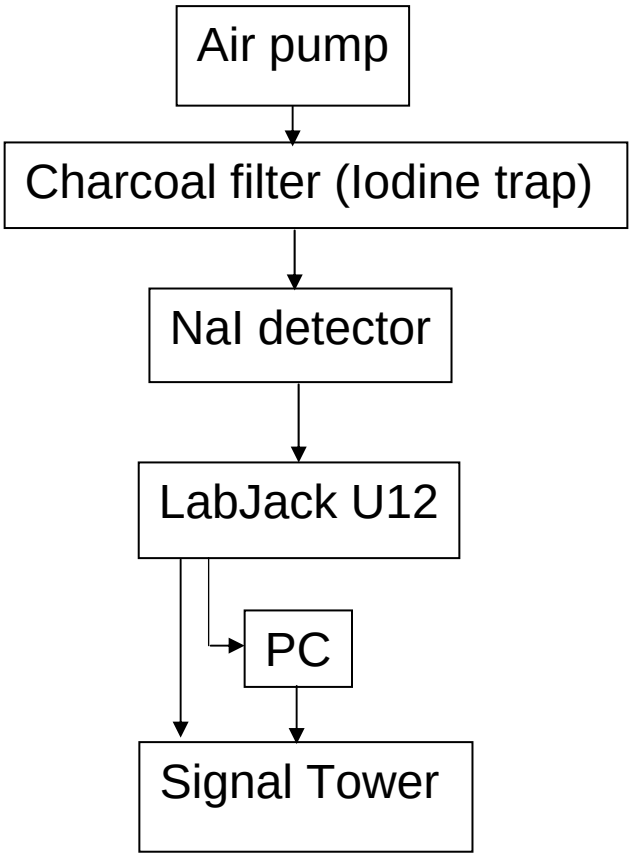


Figure 3-3-1: Main parts of the system hardware

3.1 Air Pump and Charcoal Filter

As the air of the reactor site contains iodine gas, to detect and then calculate its concentration we need to have an air pump and also a filter that absorbs the Iodine.

The pumping flow rate is stable (a typical value is 22 l/min ± 1% drift over a 3 month period). Therefore, in our analysis the air pump flow rate is approximated as a constant. It is adjustable for the operator in case something in the pump is changed.

The air is passed through the filter by the air pump. When the filter traps the I-125, the detector will notice a rise in filter activity caused by the new Iodine trapped. The filter that is used in the system is a charcoal filter (CH_3I).

Charcoal is a black, porous and carbonaceous material that is 85% to 98% carbon. It is produced by the destructive distillation of wood. It can be used as a fuel, absorbent and filter. For our purpose, trapping the I-125, a charcoal filter is very efficient. It can capture about 99% of the present I-125.

Over time, the amount of I-125 in the filter increases. This increase can make the situation difficult for the detector to detect small additional amounts of I-125. The existing iodine behaves as a large background, making it much harder to estimate how much Iodine has been trapped in the last time interval. Therefore, the filter must be changed periodically to make it possible for the detector to observe new Iodine. In the MNR it is changed every week usually on Mondays.

Figure 3-2: Air pump

3.2 NaI Detector

NaI is a scintillation crystal with appropriate characteristics for detection. The luminescence efficiency of NaI is very high. It can also be found in single crystal or polycrystalline forms in different sizes and geometries. It has fine resolution ability to X-ray and γ-ray and no important self absorption of the scintillation light. One of the most widely used materials, of all on hand scintillators, is NaI. It is used in many applications such as high energy physics, well logging, nuclear medicine and environmental monitoring. More information about the properties of NaI is available in Appendix 3.

The detector that is used for detection of I-125 gamma photons is an NaI detector. This detector has an NaI crystal mounted to a phototube (photomultiplier tube). When the incident radiation interacts in the NaI crystal, sparks of light are produced. These sparks are transmitted into the phototube. Then in this tube, the light produces electrons. The number of these electrons is multiplied by a factor of a million or more.

This kind of detector is good for low energy detection. Generally the energy range that the NaI detector can cover is between a few keV to about one MeV. The energy of iodine 125 gamma photons is about 33 keV. Thus, a special NaI detector with 1 mm thick crystal is used for this purpose. As the crystal has very small thickness, it can only detect the photons below 100keV making it less sensitive to the photons other than iodine 125's gamma.

There are three NaI detectors in the reactor: one on the experimental floor, another in the enclosure and the last one in the exhaust duct. The efficiency of these detectors is around 7%. These efficiencies have been calculated by coincidence counting and liquid scintillation methods[4].

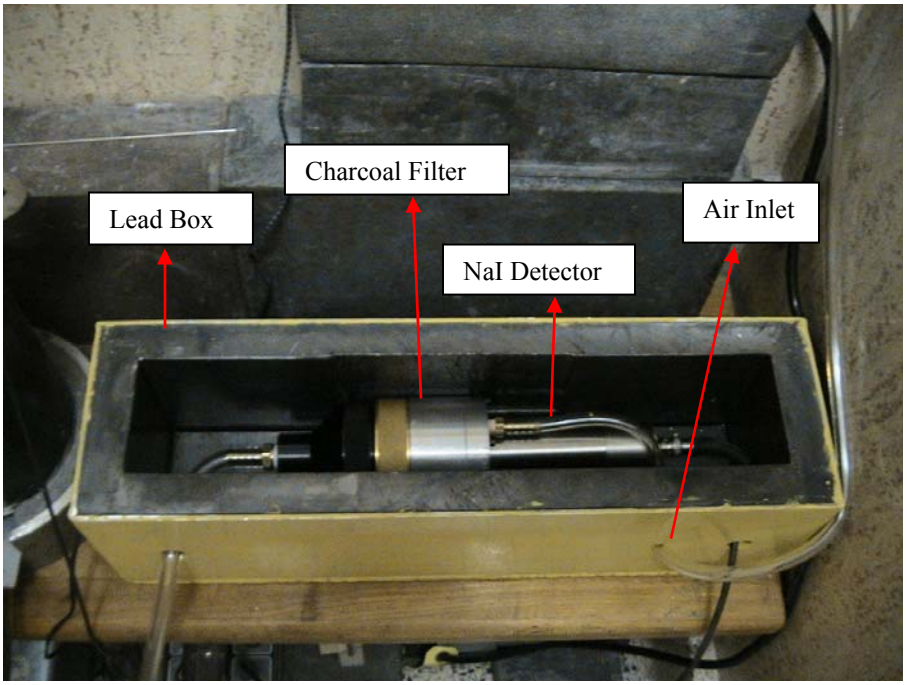


Figure 3-3: NaI Detector and Charcoal Filter in the Lead Box

3.3 LabJack U12

LabVIEW can communicate with various data acquisition boards. In MASIS, a suitable data acquisition board called LabJack U12 is used as a counter. This board also provides output signals to control the alarm and annunciation system.

LabJack U12 is a multifunction, low-cost interface that provides a reliable connection between PC and the physical world. This measurement and automation device is generally used as data logging, data acquisition, measurement, and control applications. Its connection to the PC in MASIS is via the supplied USB cable. LabJack U12 can be used with various software, such as C++, VB, Delphi, LabVIEW, VEE, DASYLab and MATLAB. LabJack does not require any external power supply. Also, it does not need any setup in the computer as Windows recognizes the board once it is plugged in. The appearance of LabJack U12 is shown in Figure 3-4.

Figure 3-4: LabJack U12

One of the LabJack's important parts in MASIS is a 32 bit counter used to find the count-rate from the NaI detector.

As seen in Figure 3-4, there are thirty screw terminals on the top surface of the LabJack. One of these terminals called CNT is the input connection of the LabJack counter. The counter is incremented when it notices when a falling edge occurs after a rising edge. This counter is able to count up to at least 1MHZ. It is read in command/response mode at up to 50 HZ.

The use of the 32 bit counter (CNT) and the watchdog timer of LabJack are mutually exclusive. Watchdog is a function that changes the states of digital I/O ports in the case of unsuccessful communication between the PC and LabJack. This state change only happens if the miscommunication continues for more than the specified timeout period. Watchdog can reboot the computer if necessary. Therefore, it provides a reliable unattended operation of LabJack. Unfortunately, when Watchdog is used, the CNT port of LabJack is disabled. For this reason we can not benefit from the watchdog function in

MASIS. To compensate for this drawback, other error handling processes are implemented in the code.

In addition to the thirty screw terminals on the top cover of the LabJack, there are 25 extra pins known as called DB25 connectors located near the USB connector on the top side of the LabJack. These are not shown in Figure 3-4.
The DB25 pinouts are listed as below:

1: D0	6: D5	11: +5V	16: GND	21: D11
2: D1	7: D6	12: +5V	17: GND	22: D12
3: D2	8: D7	13: +5V	18: D8	23: D13
4: D3	9: NC	14: GND	19: D9	24: D14
5: D4	10: +5V	15: GND	20: D10	25: D15

D0 to D15 refer to sixteen digital I/O ports. DB25 connector also has connections for ground (GND) and +5 volts.

The first four channels of digital I/O ports (D0-D3) and the channel #11 (i.e. +5V) are used for the alarm and annunciation system of the MASIS setup.

Although LabJack has some digital I/O screw terminals (IO0- IO3) at the top surface of the device, these terminals can not be used for our purpose. Each of these digital I/O ports has a $1.5k\Omega$ series resistor that provides the protection against overvoltage or short circuit accident. This resistor limits the maximum current output of each channel below which is needed to turn on the optoisolators of TriacOut4. TriacOut4 is a solid state switch between LabJack and the Signal Tower. It will be further described in the next section. Therefore, alternative ports of DB25, which do not have the current limitation of the screw terminals, are used for this purpose.

Although LabJack has different and redundant protection mechanisms, the inappropriate use of the device can damage the LabJack and even the connected computer.

Since the other features of the LabJack might be needed for upgrading the system in future, more information about this device is presented in Appendix 4.

3.4 Signal Tower

The main part of the MASIS alarm and annunciation system is a Signal Tower that includes three lights and a buzzer shown in Figure 3-5. Each light indicates different situation as follows:

-Continuous green light for normal condition.

-Continuous amber light for the warning condition when iodine release is occurring.

-Flashing red light for alarm condition that indicates approaching administrative release limit.

Figure 3-5: PATLITE Signal Tower[5]

The Signal Tower used in MASIS is from PATLITE corporation. Its model number is SEFB-302T with the following characteristics:

- The lights are LED type.
- Each light can be set to be continuous or flashing.
- The signal Tower contains a buzzer.
- All of the lights and buzzers work with 24V AC/DC.

-The Signal Tower has a steel mounting pole with L-Bracket.

The Signal Tower is mounted on the wall above the MASIS PC where all the staff can easily observe it. Figure 3-6 shows the wiring diagram of the Signal Tower.

Note that the Signal Tower has two buzzers with different sounds, but we only use buzzer 2 with skyblue wire. Therefore, we always refer to this buzzer in this manuscript.

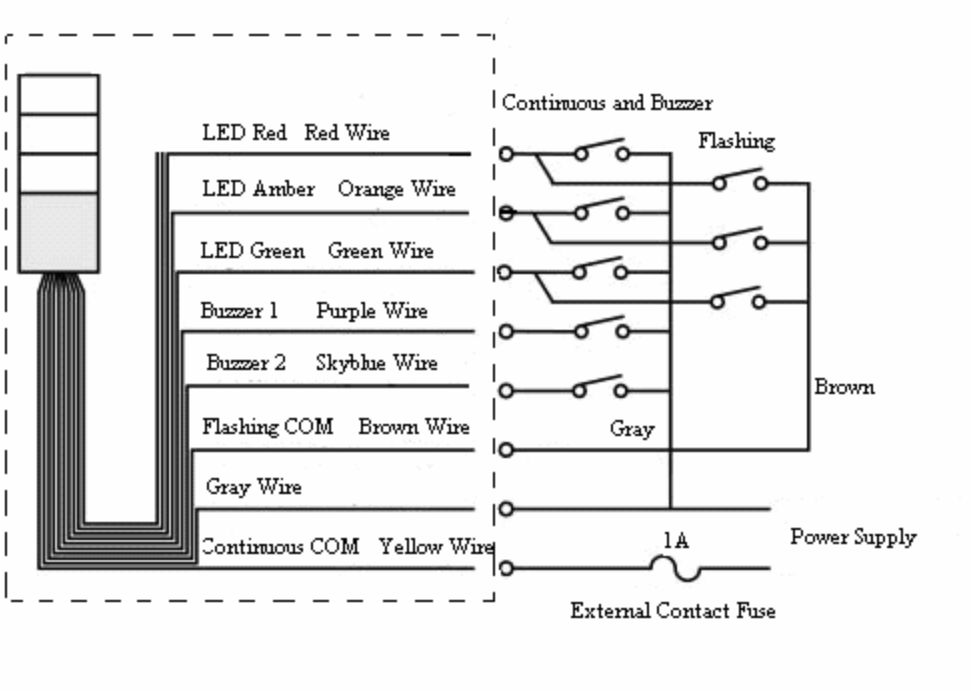


Figure 3-6: Wiring diagram of the signal Tower[6]

3.5 TriacOut4

The output voltage of the LabJack U12 is +5V with a limited small current that is not able to turn on the Signal Tower that works with 24V DC/AC. Therefore, TriacOut4[2] is employed as a switch between DB25 ports of LabJack U12 and the Signal Tower. TriacOut4 is a 4-channel triac output board shown in Figure 3-7.

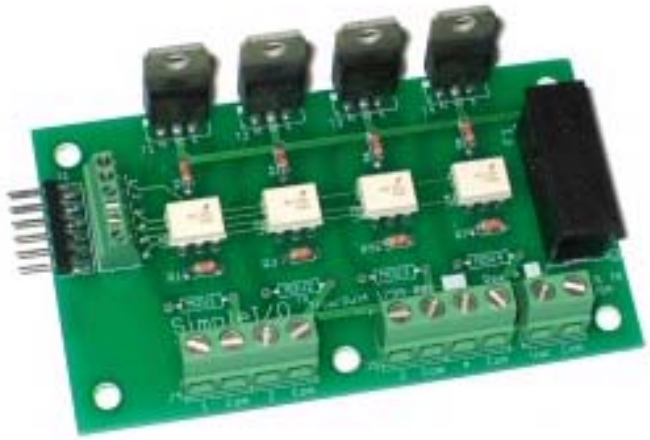


Figure 3-7: TriacOut4

TriacOut4 has four channels that make a 4-point solid-state-relay circuit for AC control applications. As shown in Figure 3-7 TriacOut4 contains:

-Triacs

-Optoisolators

-Terminal block connectors

-A fuse with snap-in cover/holder for easy replacement

The schematic picture of the TriacOut4 is shown in Figure 3-8. As it is seen there the logic input has a pin for each of the four channels. It also has a pin for a common 5V DC supply.

For each channel of TriacOut4 there is a zero-crossing optoisolator and a triac driver outputs. The triacs are derived by the optoisolators. The combination of these two, triac and optoisolator, makes a switch between the AC input and the output connector for each channel. Using an optoisolator to control the triac reduces the electromagnetic

[2] QuickSource™ inc.

interferences with the switch. All the four triacs are located on one side of the TriacOut4 board, which provides an easy connection to the heat sink.

Although each triac is designed for a maximum of 8A, the typical current limit for each channel is 1A at 120 VAC.

The AC input is a 2-pin connector. It also has a fuse. For easy replacement, the fuse has a snap-in cover/holder.

The AC output of each channel is a 2-pin connector. One pin is for the switched (hot) and the other pin is common (neutral).

Some specifications of TriacOut4 are listed as below[7]:

- 4 opto-isolated Triac AC output points
- Triacs: 8A max, typically for 1A, 120V AC on each point
- Optoisolators: zero-crossing, triac-driver, 120/240V AC typical, input +5V DC at 15 mA
- Logic input: Supply +5V DC, ground each point to enable.
- Connectors:
    1. Logic input right-angle header on 0.1 ″centers
    2. Logic input terminal block on 0.1 ″ centers
    3. AC input and output terminal blocks on 0.2 ″ centers
    4. AC outputs have hot and neutral pins for each channel
- Fuse: 6A common to all outputs, 5×20mm in a snap-in holder
- MOV footprint, for driving inductive loads.
- PC Board: 2-sided, 0.062 ″, with solder mask and silkscreen
- 3.5 ″ long × 2 ″ wide. Height of triacs are less than 1 ″.

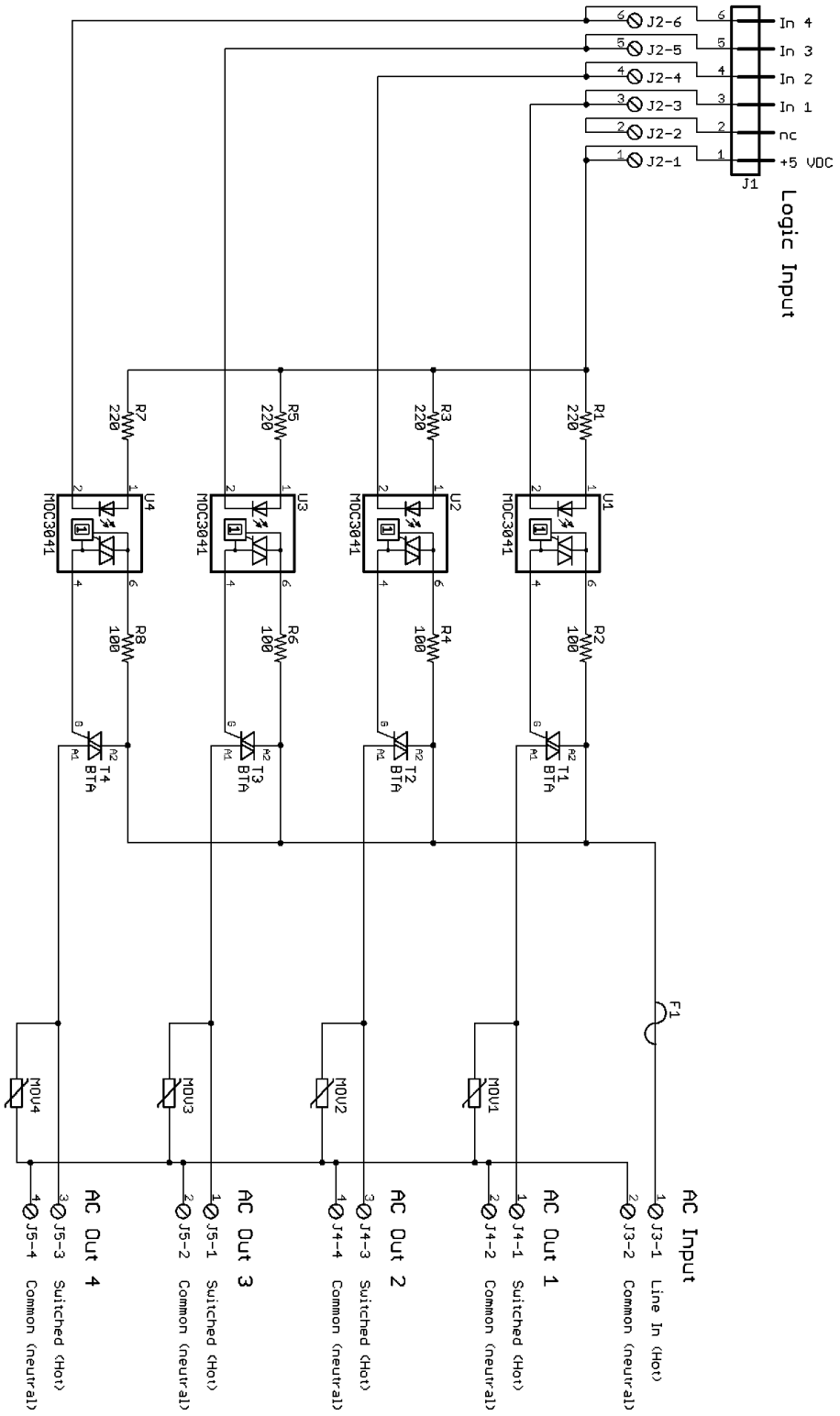


Figure 3-8: Schematic picture of Triacout4

As previously described, in MASIS the green and amber lights of signal tower are continuous, and the red light is flashing. Figure 3-9 shows that the common wire of the flashing lights (i.e. brown) is separated from the common wire of the continuous lights and the buzzers (i.e. gray). Note that these common wires are different from the common ground labeled with "continuous COM" in the figure.

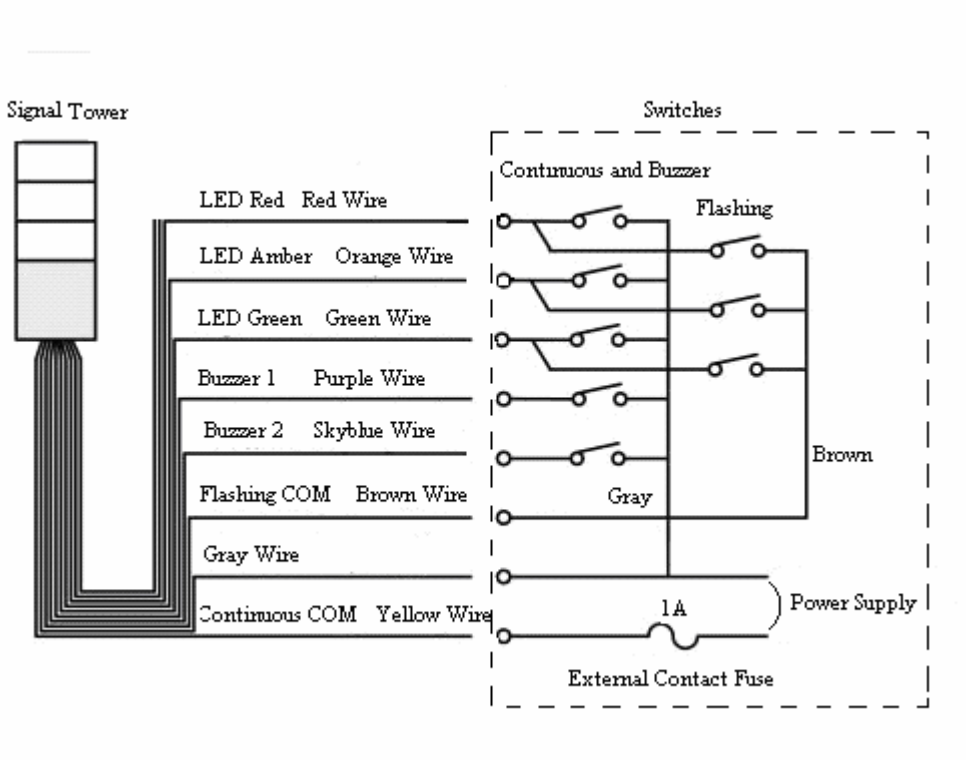


Figure 3-9: **The common wire for flashing is different from the common wire of continuous lights and buzzer.**

However, as shown in Figure 3-8, in TraicOut4 all four channels share one common wire. Therefore, in order to use one triac for the flashing red light, we will need to make a separate common wire by disconnecting the common wire of one of the channels from the other three channels. For this purpose, a small physical change is applied on the circuit of the TriacOut4, which is shown in Figure 3-10: a cupper strip is cut and a wire is soldered between two points shown in the figure. This figure also includes the labels for the connection to the Signal tower.

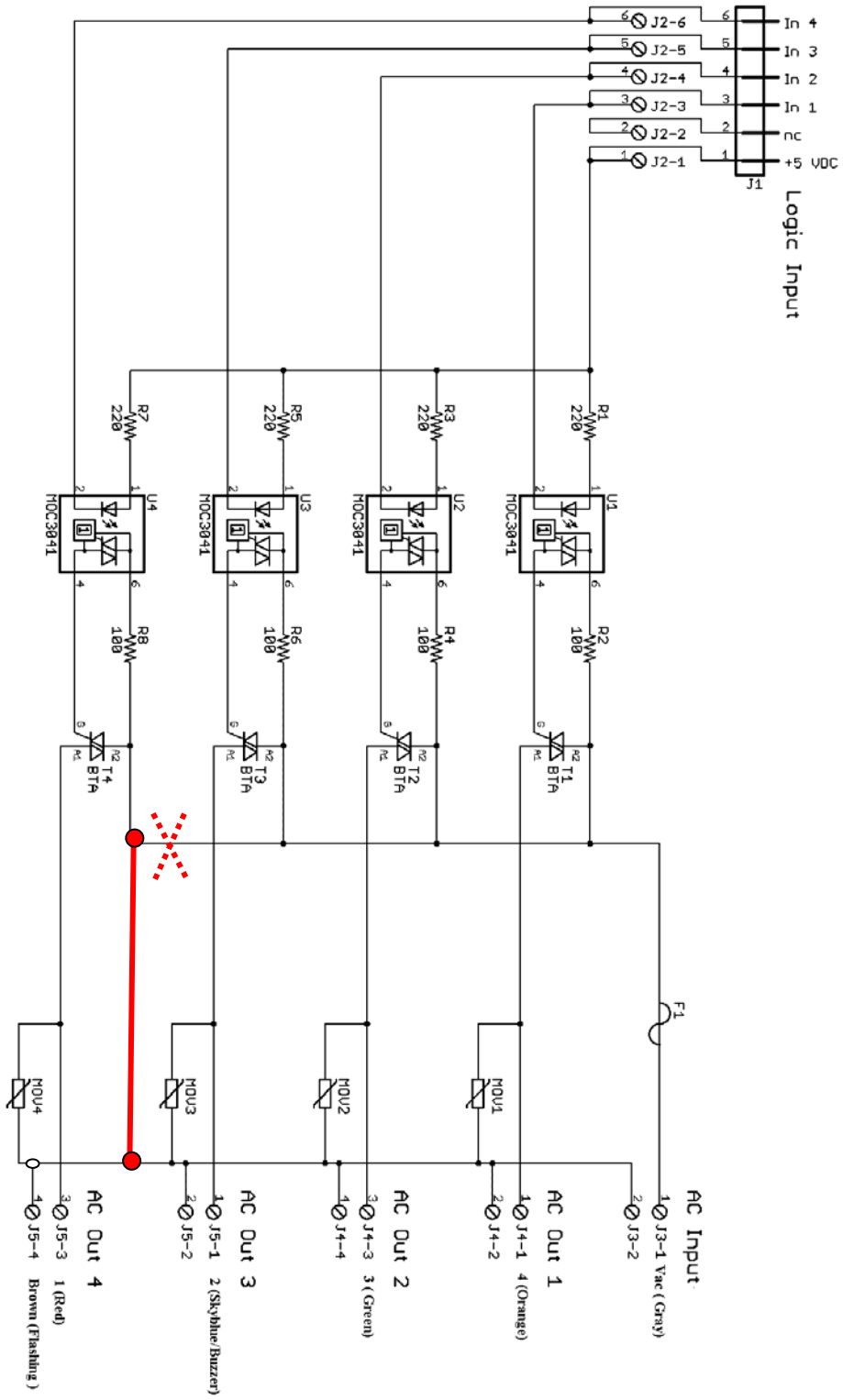


Figure 3-10 : A diagram showing the Triacout4 and the changes applied to separate the common wire of the flashing red light.

3.6 Wiring Connections

Figure 3-11 shows the wiring schematic among LabJack U12, TriacOut4, Signal Tower and the transformer. Since Signal Tower works with 24V AC/DC, a transformer is needed to convert the line voltage of 120 V AC to 24 V AC. The reason for using the 24 V AC Signal Tower, as opposed to the 120V AC Tower, is that the 120 V AC Signal Tower did not have the CSA (Canadian Standard Association) approval at the time of placing the order.

CNT screw terminal on the top surface of the LabJack is connected to the detector output to count the signals from the detector. Pins # 0, 1, 2, 3 & 11 of LabJack DB25 Connector are wired to the Logic Input of TriacOut4 board. The color codes of the wires are also labeled in Figure 3-11.

As seen in this figure the side terminals of TraicOut4 are connected to the Signal Tower and the transformer. The Signal Tower is controlled with the LabVIEW program as follows:

The program sends a signal to activate a pin in the DB25 connector. The activated pin, which has a voltage of +5 volts, turns on the corresponding triac port on the TriacOut4. As a result, the On triac connects the 24V AC power from the transformer to the desired section of the Signal Tower and turns on the lights or the buzzer.

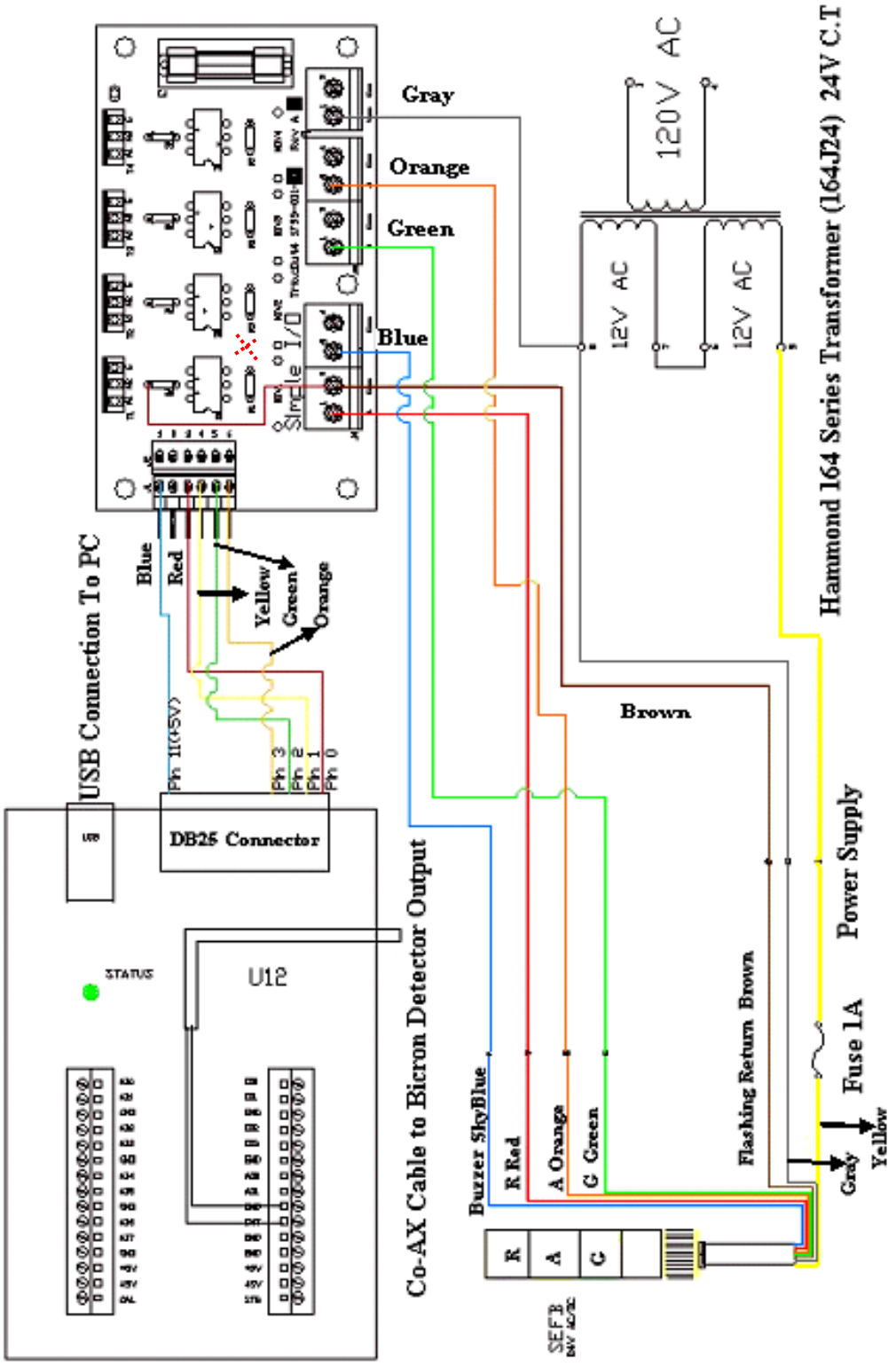


Figure 3-11: Wiring schematic between LabJack U12, TriacOut4, Signal Tower and transformer

Chapter 4

Graphical Representation of the Process in the Main Program: Flowchart

Introduction

This chapter provides an overview of the main procedures in the program and illustrates the main features by the use of flowcharts. Since the detailed flowchart of the complete program is complex and potentially confusing, herein the program is broken into its significant parts. First a top-level flowchart is provided and then each box of this flowchart is expanded with more details. In this way, a functional understanding is gained. This will provide useful means in understanding the details and complexities of the code that will be explained in the following chapter.

4.1 Overview Flowchart

Figure 4-1 shows a simple flowchart of the whole program. It illustrates four main steps. These are the four important tasks that MASIS must be able to do.

Step 1: The first task is "Counting the Signals". These signals represent the I-125 gamma photons that are detected by the NaI detector. As will be illustrated in the detailed flowchart of this part, the program counts the signals with the help of LabJackU12 that is an appropriate data acquisition board for the system.

Step2: There is an inescapable need to change the charcoal filter in the system regularly. The frequency of filter changing is usually one-week. As previously explained, this change is due to the fact that more iodine in the filter means more difficult estimation of the quantity of the new iodine arrival. Subsequently, the necessity of changing the filter in the system makes two modes of operation: "Filter Change Mode" and "Normal

Mode". In the "Filter Change Mode" the program cannot calculate the accurate Iodine Concentration. Thus it involves a particular procedure that the program should take care of during that time. Consequently, the mode of the operation must be considered prior to the calculation of the Iodine concentration

Step 3: According to Figure 4-1, the next step is calculation of the concentration of iodine. This section is the heart of the program since the Iodine Concentration is the key parameter to be determined by MASIS. All the judgments of "Alarm and Annunciation System" in the following step are based on the amount of the Iodine Concentration. Normally, the first parameter that is checked by the operators and staff is the amount of Iodine Concentration. Therefore, the appropriate demonstration of the Iodine concentration is also very important. In addition, it is needed to have a correct average of the Iodine Concentration at different time windows. All these tasks will be done in this section.

Step 4: Most of the time the amount of Iodine is normal and there is nothing to worry about. Consequently, the operator and staff may not continuously check if the Iodine Concentration is below the safe limit. It is the responsibility of the system to get the attention of the crew in case the Iodine concentration passes a specified threshold. This is achieved in different ways in MASIS:

- The current condition is directly displayed on the screen by flashing lights.
- The "Signal Tower" will show the status by different colored lights, and also a buzzer in case of an alarm.
- If the amount of Iodine Concentration reaches the alarm threshold, the program automatically sends an email to the staff. The data file is also attached to this email for quick analysis. This is especially useful if the staff are out of the building and cannot see or hear the signal tower.

According to the flowchart, each of these four steps is in a "Predefined Process" box. The definition of each box is the detailed flowchart of each step that will be presented in the following sections.

Note that these flowcharts do not include some features of the program related to the appearance of the MASIS. For instance, the process for setting up the password protection area of the code is not mentioned in the flowcharts.

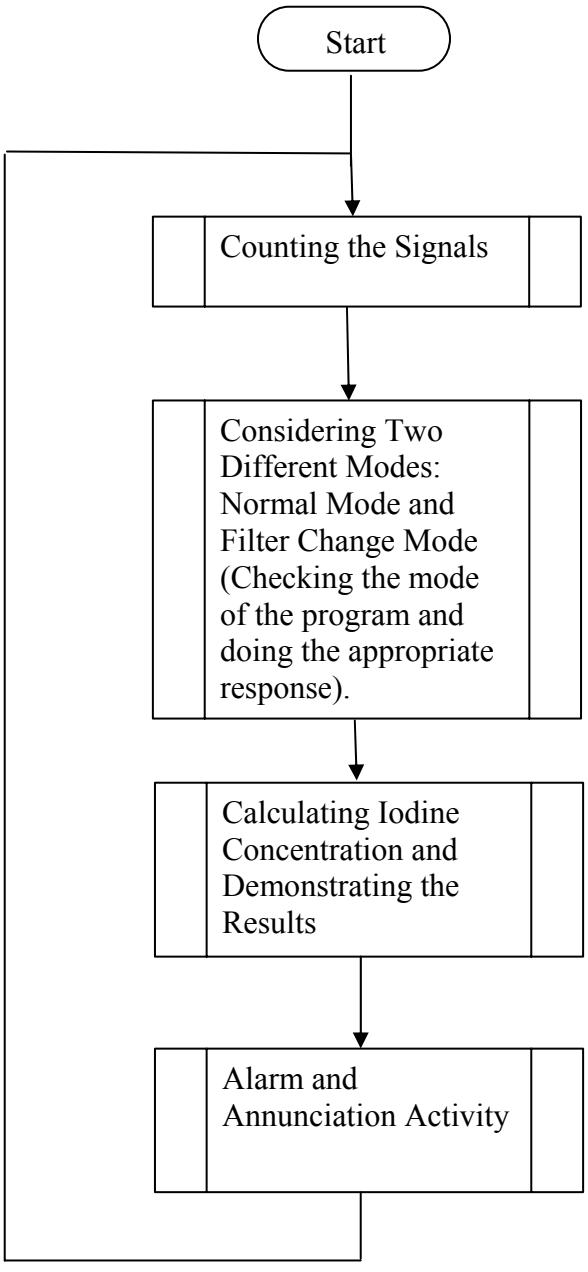


Figure 4-1: The Overview Flowchart of the program.

4.2 Counting the signals

Figure 4-2 shows the detailed flowchart of the "Counting the Signals" box in the overview flowchart described in the previous section.

As it is seen in the flowchart after "Start" at the beginning of the program, the first job of the system is communicating with the CNT LabJack U12. CNT is the port of the LabJack that is responsible for counting. This communication with LabJack, like communication with many such interfaces, may encounter some problems that need to be carefully considered. Since all the calculations in the program are based on the count-rate, its reliability is crucial. So in each iteration the program will check whether there is any error in this communication or not. If the answer is "No", it goes to the next step. Otherwise the program shows this abnormality to the staff members, and analyzes the problem to help the operator fix it. Therefore, the program will do some processes in parallel that are going to be explained here.

There is a flashing green light in the main screen of the computer that indicates the normal condition. In case of any error in communication between CNT LabJack and PC, the program is no longer in the normal operation. Hence, this green light should be turned off, which is indeed the immediate reaction of the program to such problems.

Moreover, the program is responsible to figure out what kind of problem or mistake has caused miscommunication between CNT LabJack and PC. For example, the trouble can be a mismatch between the ID number that is defined for the LabJack in the program and the actual ID of the existing LabJack. This will be explained in detail in the following chapter. Such issues should be reported so the operator can easily locate and repair the problem. The report appeared in the "set up" page of the screen in the section of "LabJack setting".

In addition to the green light at the main screen, the Signal Tower green light should be also turned off to get the immediate attention of the staff members. Therefore, the system again communicates with LabJack. But this time instead of CNT LabJack, the communication is through DB25 channel 2. DB25 is the output port used to access the Signal Tower green light. If there were no error in this communication, the green light

would be turned off with no problem and the program response to the miscommunication error between CNT LabJack U12 and the PC is finished. Now the PC can continue the communication with CNT.

But if the program encounters any problem in communicating with DB25 channel 2 to access the green light of Signal Tower, it will try to recognize and report the error. The error is shown in the related box for "DB25channel 2/green light" in the "LabJack setting" section located in the "set up" page of the screen. Then without being able to turn off the green light, the program returns to another try for communication with CNT LabJack. Unfortunately, there is a high chance that both communications with CNT LabJack and DB25 channel 2 will encounter the same kind of errors. This case happens when there is a problem with the LabJack itself, and not with the ports. In this case, the error is reported on the display and only the flashing green light on the main screen turns off. However, the green light of the Signal Tower is still "On" and the personnel who only see the Signal Tower mistakenly think that everything is alright. Therefore, until someone notices the reports on the PC and finds out that the flashing green light of the main screen is "Off", the problem will remain undiscovered. For this reason, to increase the safety level and avoid such possible ignorance, it is advised to use two separate LabJacks for communication with CNT and DB25 channel 2 to avoid common mode failure problems.

As shown in Figure 4-2, the program stays in an infinite loop if the operator does not solve the problem. The flowchart shows two choices here: either stay in the loop or the operator understands the problem and hopefully fixes it. Note that after fixing any problem the Signal Tower green light and the flashing green light of the main screen must be turned on again. To save some space in Figure 4-2, "Turn on the green light of signal tower" is in a "Predefined Process" box of flowchart. The detail of this predefined process is shown in Figure 4-3.

As it is seen in the flowchart, the green light of the screen and Signal Tower will not be "On" at the initializing period. The initializing period is one hour that follows the startup of the program and also after each filter changing. It is the time that the program

is accumulating the data required to estimate an accurate Iodine Concentration. Although the duration of it is usually one hour for both situations, the operator can adjust them in the password protected area. As in this period there is no accurate estimate of Iodine Concentration to report, the program is not in the normal operation to have the green lights "On".

After the proper communication with CNT LabJack, the program is ready to go to the next step that is "counting the signal". The counting process will be explained widely in the following chapter.

After counting, the program will check if the result is acceptable, i.e. having an appropriate communication with CNT LabJack does not guarantee a correct count-rate result. For example, assume the program continuously shows zero for the count-rate. This situation is not acceptable even if the communication with LabJack is correct. The program counts this situation as a "No Signal" error that might happen from disconnection of detector or something else in the system hardware. The explanation of error will be appeared in the "Event Log" box in the main screen.

Any day that at least one event, like the one explained above, happens, a new text file will be created. The name of the file will be "eventlog + date". In this manuscript, this is shown with "*eventlog.txt" where "*" indicates the date. It is created in a folder called "data-log" located in the local disk (C:). One of the events that creates the "*eventlog.txt" file is the count-rate error. Obviously, in the days that no particular event happens, no "*eventlog.txt" file will be created.

After saving the explanation of error and its start time and date in the file, the program will return to communicate with the CNT LabJack again and all the following procedures. As it is seen in Figure 4-2, the process exits the infinite loop as soon as the operator solves the problem.

When the program exits this loop, the reestablishment of the correct signal is reported. Therefore at this point, the program first checks whether the previous count-rate was acceptable or not. "Yes" means that everything has been alright and, as expected, the

reestablishment is not applied here. Thus the program directly goes to the task of reporting the count-rate.

But if the answer is “No”, indicating a previous error in the count-rate, the reestablishment has occurred. So it will be reported in the “Event Log” box of the main screen and also will be saved in the “*eventlog.txt” file of that day.

In the next step, the count-rate is reported by its specific indicator on the main screen. This number is updated every second and shows the number of counts at the last second.

Other boxes of the flowchart in Figure 4-2 are the same as those in the overview flowchart of the previous section.

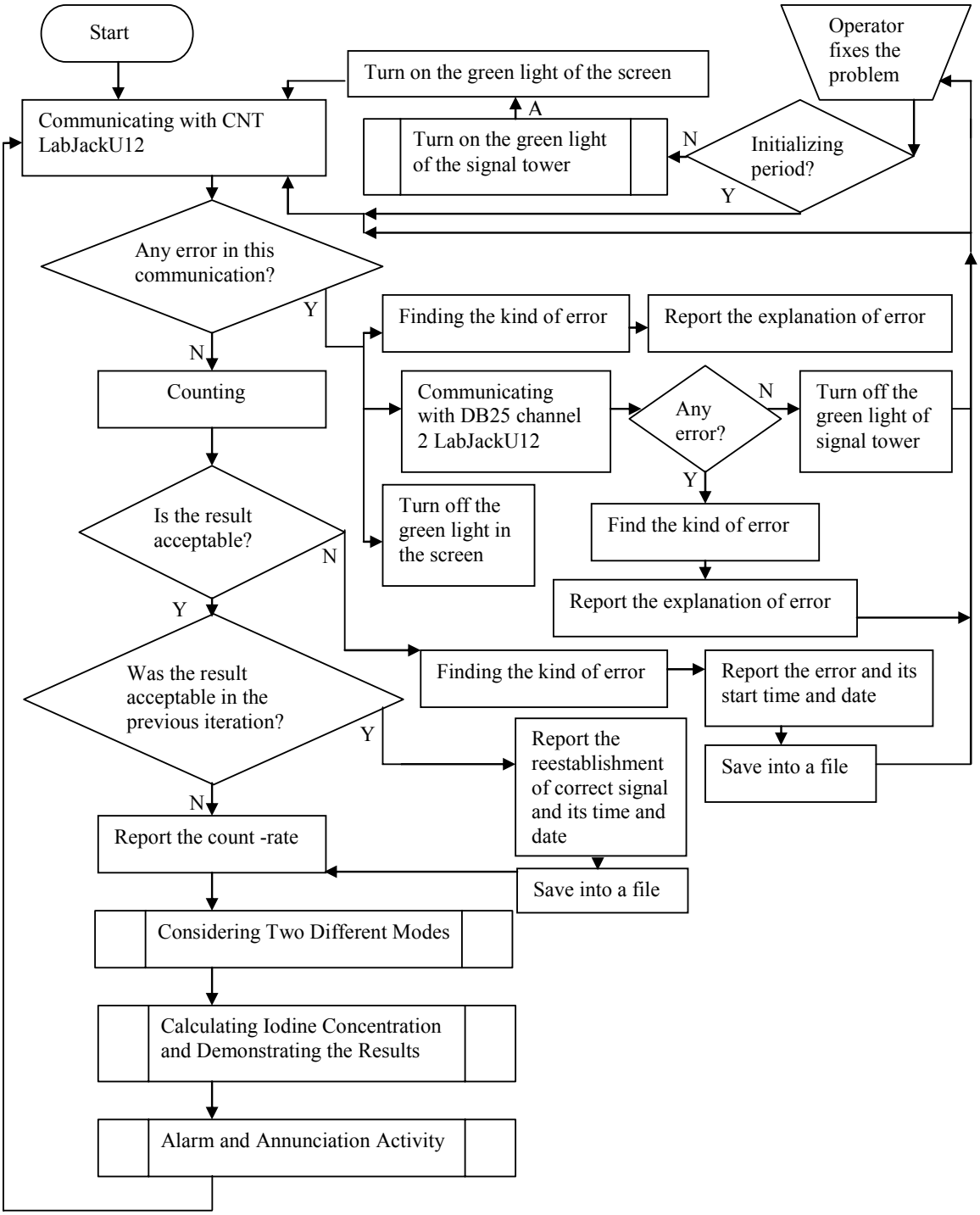


Figure 4-2: Detailed process of the "Counting the Signals" in the overview flowchart.

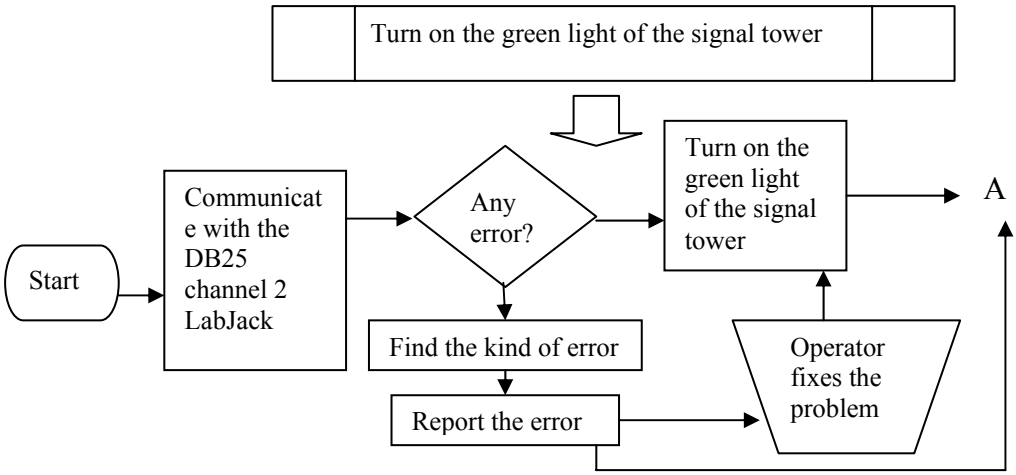


Figure 4-3: "Turn on the green light of the signal tower "is defined as above.

4.3 Considering Two Modes

Figure 4-4 shows the flowchart of the program focusing on "Considering Two Modes". This section clarifies the predefined box that indicated the two different modes in the overview flowchart in Figure 4-1.According to this flowchart after counting the signal, it is asked if the program is in the Normal Mode. If the answer is yes, without any further process it goes to the next step to estimate the Iodine Concentration. But if the program is not in the Normal Mode, it is not allowed to use the current count rate for the Iodine Concentration estimation. As there are only two modes in the program, when it is not in the Normal Mode, the Filter Change Mode is reported. Arriving to this mode is reported in the "Event Log" box of the main screen. Also it is necessary to save the time and date of entering to the Filter Change Mode. It is saved in the "*eventlog.txt" file of that day which was explained before.

Also, as the program is in the normal mode, the flashing green light on the main screen and also the green light of the Signal Tower, which are the indications of normal operation, are "Off". In order to turn off the green light of Signal Tower, communication with a specific channel of LabJack is necessary. This task is set in a predefined process

box. The definition of this predefined process is similar to that of "Turn on the green light of the signal tower" in Figure 4-3. There is only one difference in one of the boxes: the box of "Turn on the green light of the signal tower" is changed to "Turn off the green light of Signal Tower".

Actually entering into the Filter Change Mode is like a semi-pause in the program. If the program does not return to the Normal Mode, it stays in an infinite loop and neither calculates the Iodine Concentration nor continues other processes.

There is a toggle on the "set up" page of the screen for setting the mode of the program. Since it is of crucial importance not to change the mode of operation by mistake, when the operator changes the mode to Filter Change Mode, the program shows an immediate notice indicating change of mode. This notice is a dialog box designed to pop up in the screen right after arriving into Filter Change Mode. The dialog box also reminds the operator to change the mode when s/he is done with changing the filter because forgetting to go back to normal mode is very critical as well.

In the Filter Change Mode the system hardware is obviously not complete and the system can not accumulate any data. Therefore, it can not be at the initializing phase. Even if the mode changes to Filter Change Mode at the middle of the initializing phase, the program exits that phase. Hence the indicator of initializing period on the main screen is always disabled and grayed out at Filter Change Mode.

Another important task that the program performs right after arriving into this mode is the calculation of the total average Iodine Concentration during the Normal Mode of the old filter. First, asked if this is the first time after running the program that the filter is being changed. If "No", this average is computed from the end of the previous Filter Change Mode until the current time. If "Yes", obviously the average is calculated from the beginning of the program. Since the Iodine Concentration average of each filter is an important parameter that should be kept as a record, it is saved into a file called "writeweekly.txt". The data of this file is loaded into a graph called "Long Term Average I-125 Concentration (Bq/m^3)".This procedure will be explained extensively in the following chapter.

After all the procedures above, the program checks the mode again. If it is not yet returned to the Normal Mode, it measures the time passed since arriving to the Filter Change Mode. Note that changing the filter usually takes ten to fifteen minutes. If the time shows more than this, there is a possibility that the operator has forgotten to set the program to the Normal Mode. Therefore, if the duration of staying in the filter change mode reached to twenty minutes, a dialog box will open and reminds to change the mode. But if the interval time is not more than twenty minutes, without opening any dialog box, the program directly returns to check the mode again. It will stay in this loop until it goes back into the normal mode.

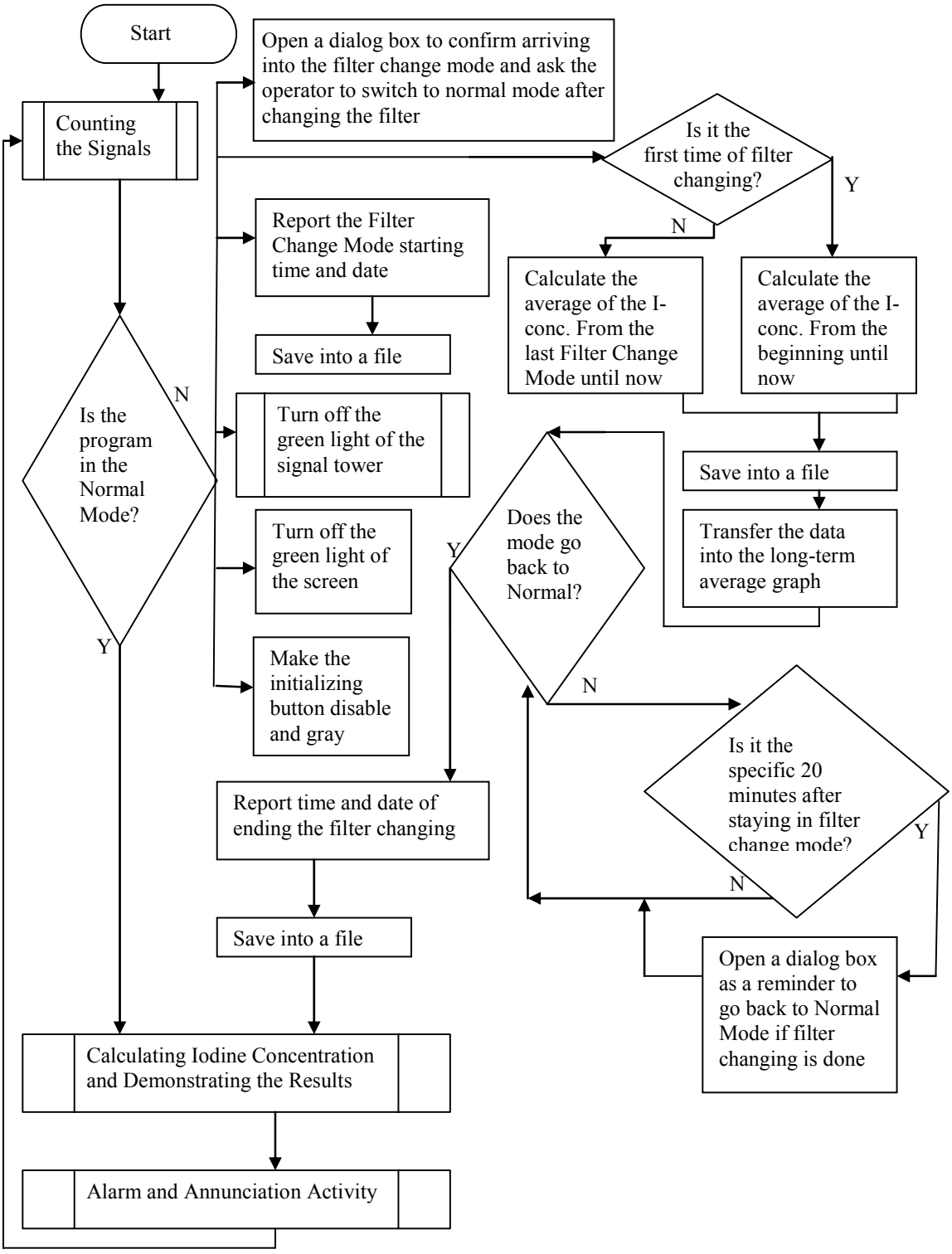


Figure 4-4: Details of the "Considering Two Modes" is shown in this flowchart.

4.4 Calculating the Iodine concentration and Demonstrating the Result

Following the flowchart of Figure 4-5 , after "Considering Two Modes", the Iodine Concentration will be calculated and its error is estimated at the subsequent step.

There is now a question box determining whether the program is at the initializing period or not. As previously explained, initializing period is a specific time to collect data after startup of the program or after the filter change.

The reason for separating the treatment of the initializing period lies on how the Iodine concentration is estimated. In order to find the Iodine Concentration, the program needs to have enough raw data from the previous count-rates. More data result in a more accurate estimate of the Iodine Concentration. The data that are gathered in one hour by the program can produce an acceptable estimation. After this initial period of one hour, the program always uses the available data from the previous one hour to estimate the Iodine Concentration. Obviously, during the initializing phase, the program does not find enough data to use for the estimation of the Iodine Concentration and the Iodine Concentration is not accurate at that time. Practically, it uses the maximum amount of data that can be found at that stage.

The operator can adjust the duration of the initializing period, if she/he is in rush and needs to get the result quickly, but this results in a higher error in the Iodine Concentration. Or she/he may increase the initial time of the data sampling to improve the accuracy of the estimations. At any rate, the program does not show the amount of the Iodine Concentration during this period and does not record them since these data are not reliable.

To show the observer that the program is at the initializing period, there exists a flashing indicator on the main screen that is enabled at this time. The program also records the starting time of this period into a file for the future data analyzing.

As during the initializing phase the program cannot calculate the concentration of I-125, this phase is not assumed as normal operation. Therefore the green light of the signal tower and also the flashing green light of the screen are turned off.

After all these processes, the program will again check to find out if it is still in the initializing period. If it is, the program will stay in the loop and ask again until ending this period. Then it will save this time into a file and go to normal progression.

After ending the initializing period, first of all, the initializing indicator on the main screen will be disabled and grayed out. Also the accurate estimation of Iodine Concentration is reported on the main screen. It is updated at each second and saved into a file.

Although, the error of Iodine Concentration is calculated at all times, even if the program is in the initializing period, it is only reported and recorded into a file after the initializing period. Obviously, because of the inaccurate estimation of Iodine Concentration in the initializing period, the error will be very high and unreasonable at this stage.

The reliability of each Iodine Concentration data is related to its percentage of error. Such information will be also used for future data analysis. The error is small when the filter is new. As the amount of iodine is increased in the filter over time, the detector capability in detecting the new iodine trapped by the filter is reduced. The percentage of error will be larger then. At this moment that the error is so large that the Iodine Concentration estimation is not reliable any more, the filter must be changed. By experience it is found that one week is a typical age of a filter.

The running average is another important parameter that is calculated. It is the average of the Iodine Concentration from ending the initializing phase to the present time. As after each filter change the program has an initializing phase, it can be said that running average is the average of Iodine Concentration in the current filter until the present time. Running average is reported in a specific box on the main screen.

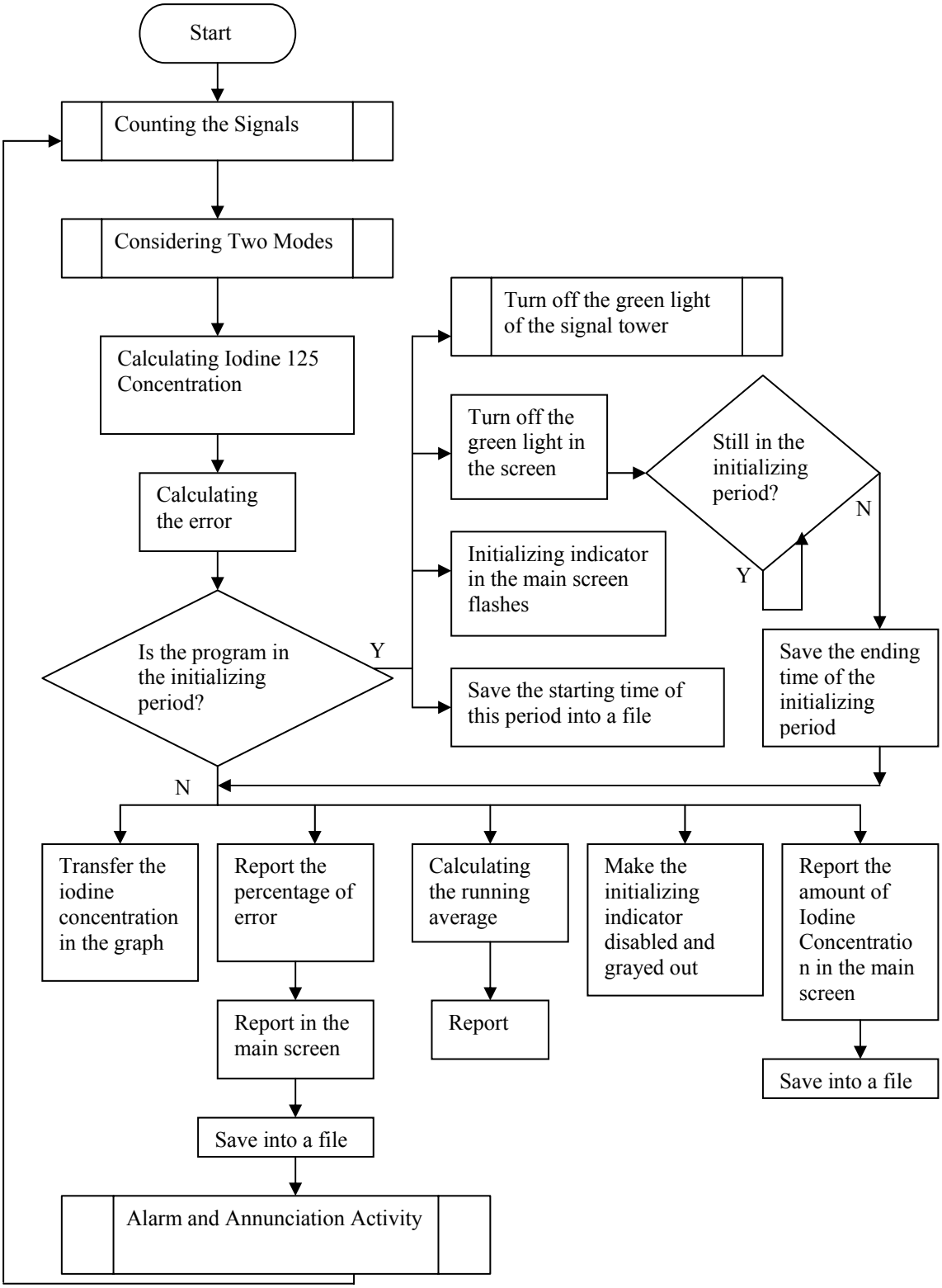


Figure 4-5: The detail of "calculating the Iodine Concentration and demonstrating the result" in the overview flowchart.

The amount of the Iodine Concentration versus time should be displayed in a graph. This is the focal graph of the program in the main screen that gets the first attention of the observer. During the initializing period nothing transfers to this graph and it will be blank at this time.

After all these processes the program will go to the next step that is the "Alarm and Annunciation Activity". This step is going to be explained in the next section.

4.5 Alarm and Annunciation Activity

Figure 4-6 shows the detail of "Alarm and Annunciation Activity" box in the overview flowchart of the program in Figure 4-1 After calculation and demonstration of the Iodine Concentration, the program's job is to properly manage the alarm and annunciation activity based on the amount of Iodine Concentration.

As explained before, there exists a Signal Tower in the system that has three lights and a buzzer for different situations. Each of these lights turns on if the Iodine Concentration exceeds its specified threshold level. First of all, the program checks if the amount of Iodine Concentration goes over the threshold set for the green light. The green light indicates the normal condition. If the program is in normal condition, this light in the Signal Tower is turned on. This task, "Turning on the green light of the Signal Tower", is set in a predefined process box in the flowchart of Figure 4-6. It was defined in Figure 4-3. Also, for the normal situation there is a flashing green light in the main screen that mimics the green light of the Signal Tower.

But before turning the green lights on, the other lights, amber and red, and also the buzzer must be turned off. There is a possibility that these lights and the buzzer are "On" from the previous iteration. These lights are digital, i.e. are either "On" or "Off". Also, they do not turn off automatically if the Iodine Concentration falls below the threshold, and the program has to turn them off if this happens. This task is shown in a predefined process box. If all the lights and the buzzer are already off, this assignment does not affect the program. For amber light and red light the program needs to

communicate with channel three and channel zero of DB25 respectively. To control the buzzer the communication with channel one is required. For more detail, the predefined box is shown in Figure 4-7.

As illustrated before in the previous flowcharts, in case of any abnormal situation, like any error in the system, the green light will be "Off". Also it is turned off during initializing period and Filter Change Mode. At these phases, initializing period and Filter Change Mode, the program is in an infinite loop and never reaches to the part of "Alarm and Annunciation Activity" and the lights are all off.

If the program is in the normal operation situation, after turning on the green light of the main screen and the Signal Tower green light, it returns to the first step of the program for counting the signals.

Now assume that the amount of the Iodine Concentration exceeds the threshold for the green light. At this state, the program verifies if this amount is in the range of amber light. If yes, first the green light, red light, and the buzzer are turned off in case they had been "On" from the previous iteration. Then the amber light will be "On", both in the Signal Tower and in the main screen. The process of turning on the amber light is like turning on the green light as shown in Figure 4.3 .The only difference is that for the amber light the program needs to communicate with channel three instead of channel two for the green light. The amber light indicates the "Warning" situation that means "Iodine Release Occurring". After turning on the amber light the program returns to the beginning of flowchart to count the signals.

But if Iodine concentration exceeds the green light threshold and also is not in the range of the amber light, it is obvious that it is in the higher range and reaches the red light threshold. In this situation, the program turns off the other lights first. Then it turns on the main screen red light and the red light of Signal Tower. These lights are both flashing. When Red lights are "On", the program is in the "Alarm" state that means "Approaching Administrative Release Limit".

To avoid distracting the personnel who are responding to the situation, the operator can cancel the buzzer from the display by pushing the "Silence the Audible

Alarm" button in the main screen. But if the "Alarm" condition continues, the buzzer will be automatically re-activated after a specific period of time. This duration is usually ten minutes but can be adjusted by the operator in the password protected area.

According to the flowchart, before turning on the buzzer of the Signal Tower in the "Alarm" situation, the program checks if the audible alarm is silent (i.e. the operator has canceled the buzzer). If not, the buzzer will be operated. If yes, the program records the time from canceling the audible alarm to the present time. If this time exceeds the specific period of time that has been adjusted to keep the audible alarm inactive, i.e. usually ten minutes, the cancellation of the buzzer will be removed and the audible alarm will be re-activated.

Also after turning on the red light, the system sends an alarm email message to the staff of the reactor whose email addresses have been listed in the recipients. Obviously, the program does not send this email at all iterations. In order to do this task, the program checks the last time of email sending. If the specific time is passed and the program is still in the "Alarm" situation, another email will be sent. This specific time is usually one hour, but the operator can adjust it in the password protected area.

After all of these activities in "Alarm" state the program returns to the first step for counting the signal.

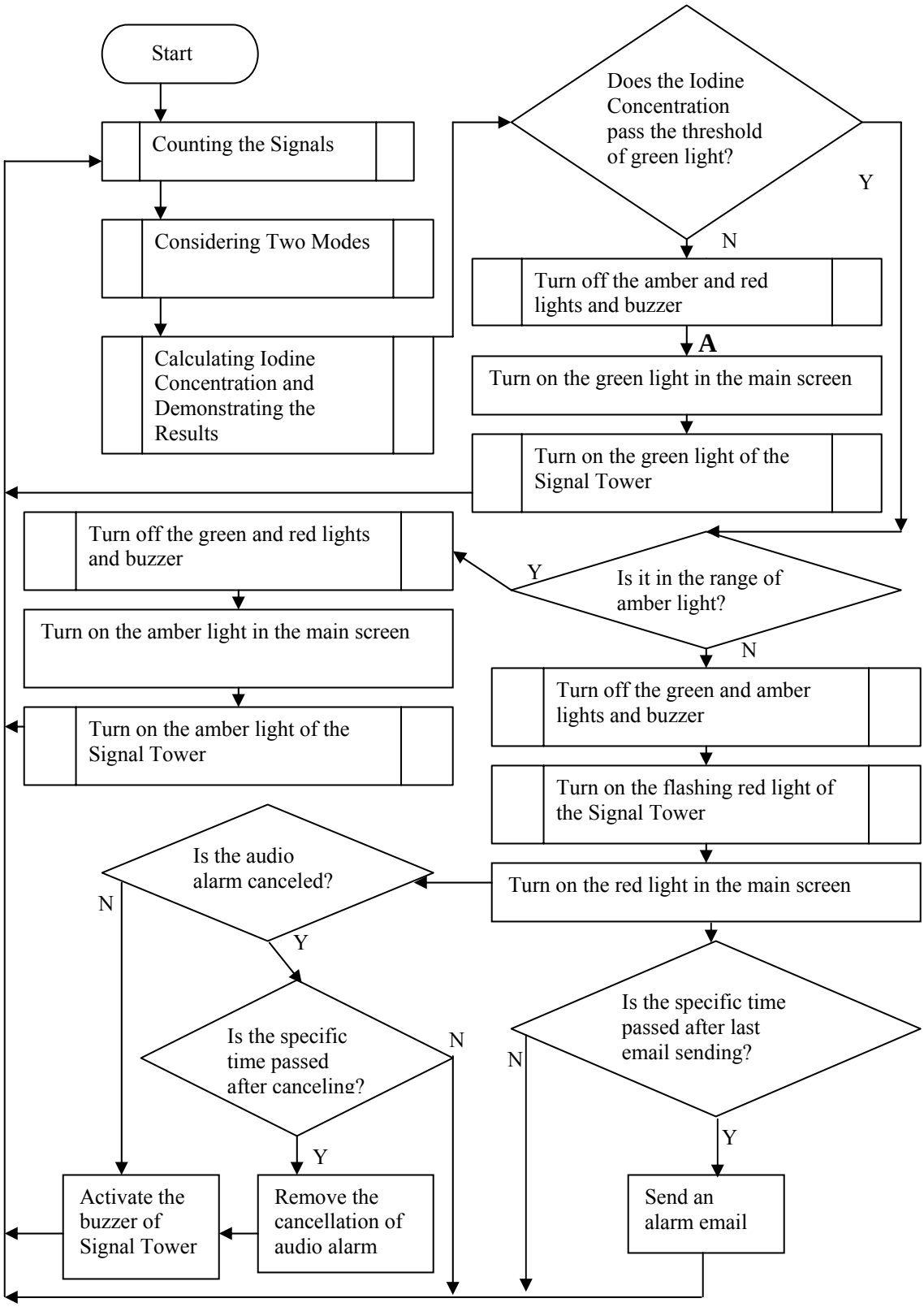


Figure 4-6 : The flowchart of the Alarm and Annunciation System in the overview flowchart.

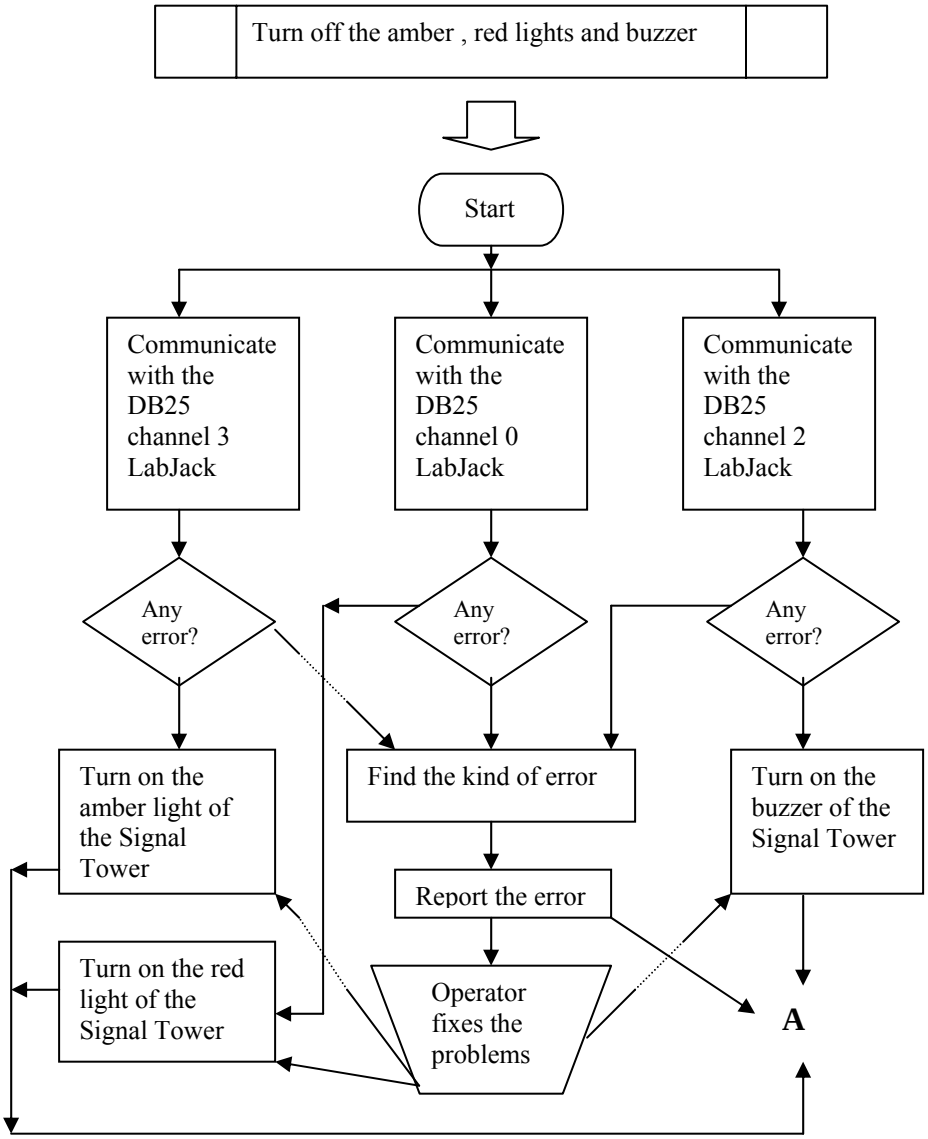


Figure 4-7: Definition of "Turn off the amber, red lights and buzzer"

Chapter 5

Code Description

Introduction

One of the problems of the previous Iodine Measuring System was the lack of access to the source code. This problem makes the program impossible to upgrade or migrate to other environments. Lack of access to the source code also causes many difficulties for the maintenance. Therefore, for the new system it is planned to not only make the code quite accessible, but also prepare a detailed description of the code. This will be the most valuable help for the next person who wants to upgrade the program or add some features to it.

This chapter is prepared for this purpose. The explanations are at the level that a person with primary knowledge of LabVIEW is able to follow. Also, all functions and sub-VIs are enlightened so in future one can use them with detailed understanding, and not just as black boxes.

After a brief description of LabVIEW at the beginning of this chapter , in the next four parts, the procedure of the code follows what was explained in the four main sections of the previous chapter, where a detailed description of the flowcharts were presented. These parts contains: “Counting the Signals”,” Considering Two Modes”, “Calculating the Iodine Concentration and Estimating the Error” and “Alarm and Annunciation Activity”. Next three sections of this chapter include parts of the code that are related to the execution and appearance of the program: “Creating files, writing into and reading from them”, “Plotting the graphs” and “Making and Controlling the Password Protected Area”.

5.1 LabVIEW

The capability of personal computers in computations and analyzing data causes extensive changes in the field of instrumentation. Modern instruments take advantage of the computer's processing power to present greater performance. A computer and also a graphical software unit with different interfaces are often used to communicate with hardware and control it.

Progressively, graphical programming software showed themselves as convenient programming tools. In these software, users only need to operate the objects on the computer screen and do not need to remember the code scripts. Despite the fact that, at the beginning, graphical programming languages could not compete with the usual and traditional programs such as C or C++, they eventually showed an environment that was quicker and simpler to employ.

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is an intuitive graphical programming software that is one of the fastest available graphical environments.[8] LabVIEW with many capabilities for designing an efficient system is revolutionizing industry. It is used in developing programs for simulation, data acquisition, control, automation, and communication applications. It was developed by National Instruments Corporation in 1986. Since then, by using these graphical developments, enhanced quality, shorter time to market, and superior engineering and manufacturing efficiency has been achieved. In other words, the time and money is saved while having a better design.

LabVIEW has many built-in measurement and control functions for different applications. It can acquire, measure, analyze and present the signals without the complexity of traditional development tools. It can connect to different measurement instruments and acquire signals and gather data easily. LabVIEW with more than 500 built-in functions can take out meaningful information, process signals and analyze the data that is gathered and acquired from the connected devices. For presenting data, there are many facilities in LabVIEW for visualizing and organizing data, web publishing and generating reports.

With LabVIEW we have this distinctive facility to develop once and set up a wide variety of computing targets from desktop to an arbitrary 32-bit microprocessor. This gives the user the most flexibility to design a system.

LabVIEW desktop format compatible with Windows, Mac and Linux is the most popular format of LabVIEW and the majority of users employ LabVIEW with their PC, Laptop and industrial PCs.

Some applications may need portability. LabVIEW PDA module can expand the LabVIEW applications to handheld devices that are running Microsoft Pocket PC 2003, Palm OS, and Windows CE.

To have extra reliability, usually for industrial monitoring and control applications, the LabVIEW Real-Time module can be used. Also with some LabVIEW products, directly programming FPGAs, DSPs, and microprocessors is possible.

The most popular advantage of LabVIEW is its ability for graphical programming. The program that LabVIEW creates is called Virtual Instrument (VI). Virtual Instrument can do many tasks, such as obtain and analyze data, use graphs and charts for displaying processed data, control external instrument and execute simulation.

One with some knowledge of graphical tools and no knowledge of programming languages like C++ can create a VI. In this graphical language, there are icons that are interconnected to make a program referred as a VI or Virtual Instrument. The extension of all LabVIEW programs is ".vi".

In the past, the possibility of changing the functionality of an instrument was never accessible and the vendor or the manufacture controlled the functionality of instruments. As Virtual Instrument is a software file, we can simply reconfigure and prepare it for other arrangements.

All VIs have two windows: Front Panel and Block Diagram. For programming, the function icons are wired in the block diagram. But the controls, indicators and switches are in the front panel. The result of the coding in the block diagram appears in the front panel.

In MASIS, we have taken advantage of LabVIEW 7.1. When the code writing was already started LabVIEW 7.1 was its latest version. At the time it was almost finished, the next version of LabVIEW, LabVIEW 8, was published. The code can be opened in LabVIEW 8 and the new features of the new version are not needed in the code. Hence, no point is seen in changing the version that has been used.

5.2 Counting the Signals

In this section, two techniques for counting the signal are discussed. Firstly, LabJack U12 is used as an appropriate data acquisition board for the system. In a second approach, the soundcard of the PC is used. Although an extensive code was written to count the signal using the soundcard, it did not give a reliable count-rate. In this section, only the main part of the code is discussed very briefly in order to explain the problems that one would encounter if such a technique is used.

5.2.1 Using LabJack U12 to Count the Signal

As mentioned before, for the purpose of counting the signal (and some other applications) a data acquisition board called LabJack U12 is employed in the system. It has a 32 bit counter with a screw-terminal input connection called CNT. To communicate with this counter, there are some sub-VIs or functions available in LabVIEW like “Ecount (easy function)”, “AOUpdate”, and “Count”. Since they all have the ability of resetting and reading the counter, they can be used to communicate with CNT LabJack. In MASIS, the ”Count“ is employed as the counting function. The appearance of this function is shown in Figure 5-1.

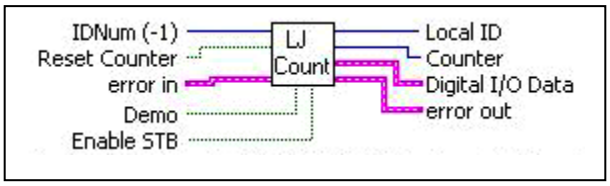


Figure 5-1: The function of "Count" for communicating between the CNT port of the LabJack and the LabVIEW code.

"IDNum(-1)" is the input serial number of the connected LabJack. If there is only one LabJack connected, we do not need to set the "IDNum(-1)" and can leave it disconnected. In this case, its value is by default equal to "-1", indicating the connection of the first found LabJack.

"Reset Counter" is another input. If it is set to "True" constant, it resets the counter to zero after being read.

"Demo" is an input that if is set to "True" constant, the counter generates random values. This can be used to run and test the program without LabJack. Otherwise, if the program runs without a connected LabJack, it generates "No LabJack" error. This error would be reported in the LabJack setting section on the "set up" page. In the "Demo" status, the program gives us a demonstration of an unreal counting, resulting in a very high percentage of error.

"Enable STB" can be used to make this port of LabJack (STB) active. STB is used for testing and calibration.

Among the Outputs, "Counter" gives us the value of CNT port of LabJack before resetting. This output is the key output of this function.

"Local ID" returns the serial number of the LabJack that counts the signals. "Local ID" is "-1" if the default value is used in the code.

The output of "Digital I/O Data" states the I/O ports of the LabJack.

By setting the "error in", the errors occur in the function of "Count" is reported in the output of "error out".

The code of sub-VI "Count is shown in Figure 5-2 and the corresponding front panel is in Figure 5-3. These figures let us know how this important sub-VI of the

program is able to do its job. The execution of it takes up to 20 ms and provides a maximum update rate of about 50 Hz.

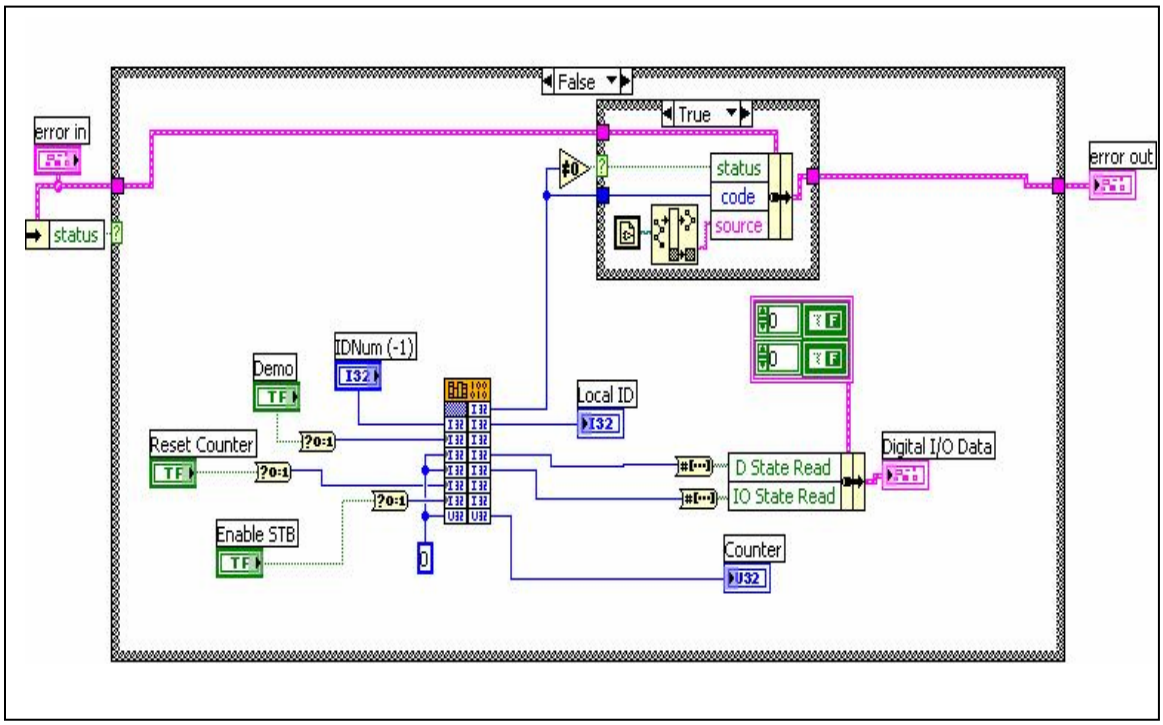


Figure 5-2: Block Diagram of the sub-VI "Count". Other cases of the case structures are empty. They are just connections between "error in" and "error out".

An increment in the number of counts happens when the counter of the LabJack detects a falling edge followed by a rising edge. Therefore, the program may not have an accurate number for the first count. This situation occurs when the program resets the counter while it has a low signal. As a result, until the signal goes high-low-high, the counter cannot show the first count. In such cases when the first count is very essential, we should not do the reset in the code but it is better to subtract the initial count from the final count instead.

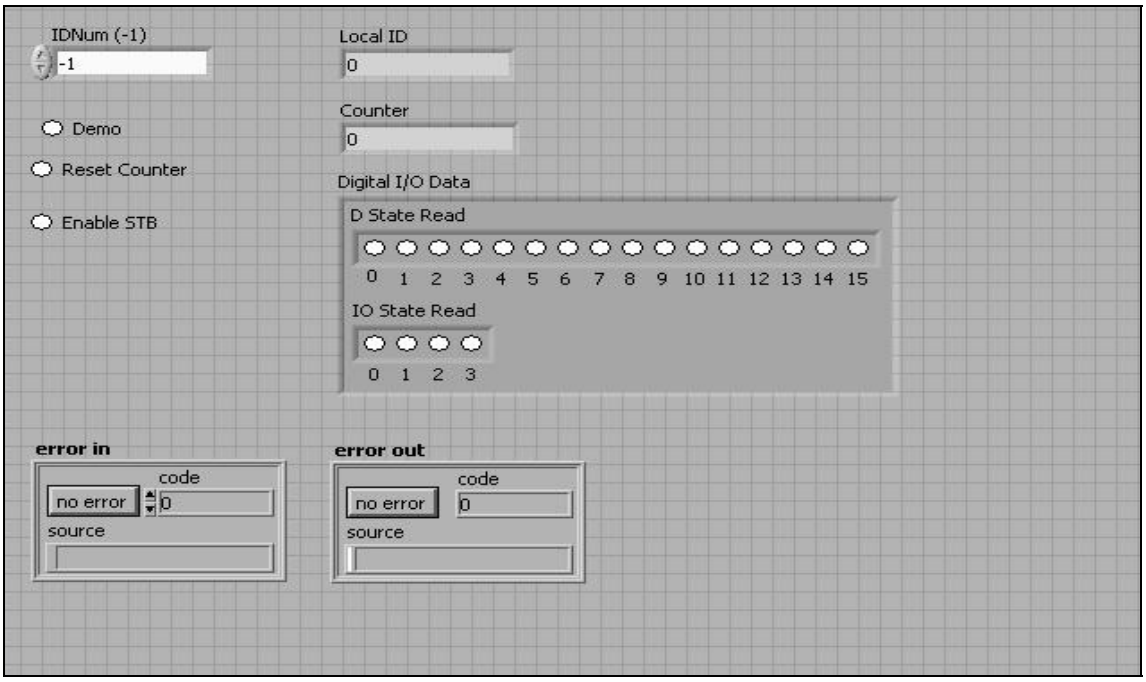


Figure 5-3: Front Panel of sub-VI "Count".

Figure 5-4 shows how the function of "Count" is employed in the MASIS. This figure only shows the main part of the counting process. Other parts are scattered on different places in the code. This is a common arrangement in this chapter when the code of a specific task is shown. For example, different sequences of stacked sequence structures usually share their area with other tasks. Due to the space limitation; it is not practical to show all the details of a specific task from different sequences of stacked sequence structure at one place. Therefore, only the main part that contains the key functions of the task is usually shown in a figure and the necessary details, which can not be shown there, are discussed in the text.

For the estimation of the Iodine Concentration with an acceptable accuracy, the count rate needs to be known at each second. Therefore, the LabVIEW code reads the count rate at each second. However, as an extra option, as seen in Figure 5-4, the time interval for the counting can be adjusted to any value. Since it rarely happens that the operator needs to change this time interval, also to avoid crowding the Front Panel, this

option is hidden and the operator cannot change the interval without having access to the main code.

Infrequently, after having a new filter, the background of Iodine in air might be larger than the background of it in the filter. In such cases the program may have a negative value for the count-rate. Thus, the code should ignore these numbers for calculating The Iodine Concentration.

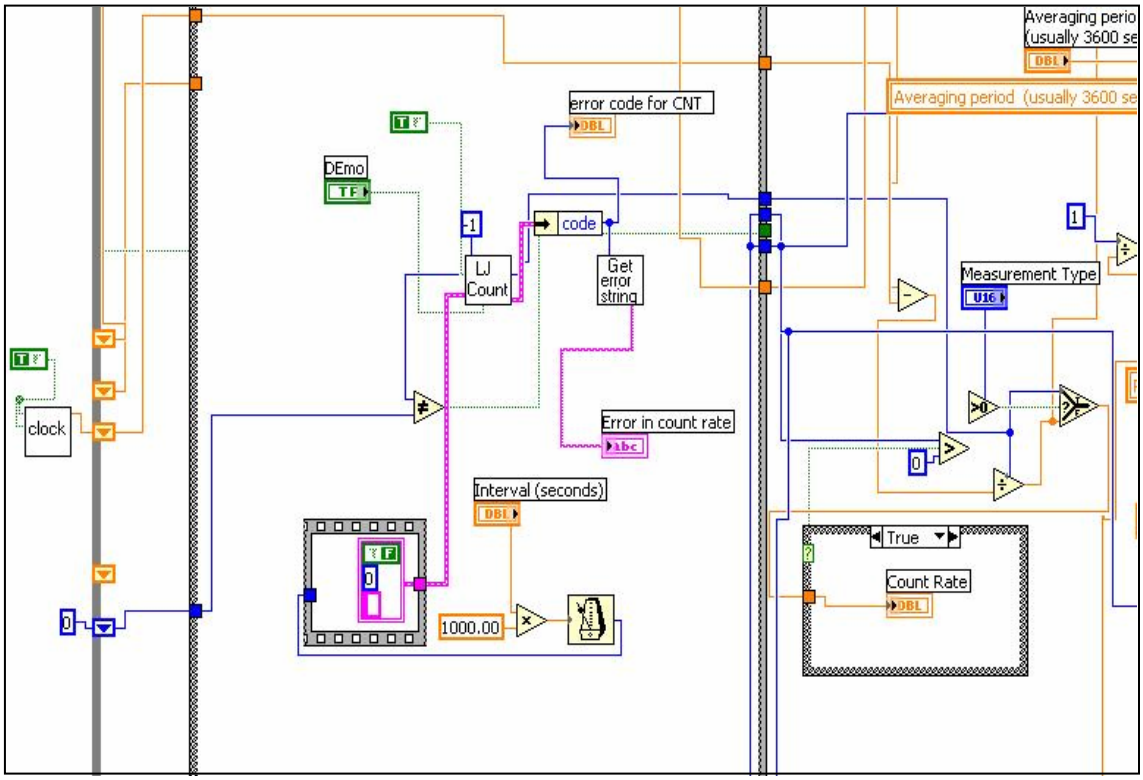


Figure 5-4: Main part of counting section in the MASIS code.

When the counter does not receive any signal to count, the program shows the "No Signal" error in the "Event Log" box.

Figure 5-5 shows the code of this procedure. As seen in this figure, the program ignores the first ten iterations. This is to make sure that such situations are not temporary and will continue.

The most probable reason for a loss of signal error is a loose connection in the system.

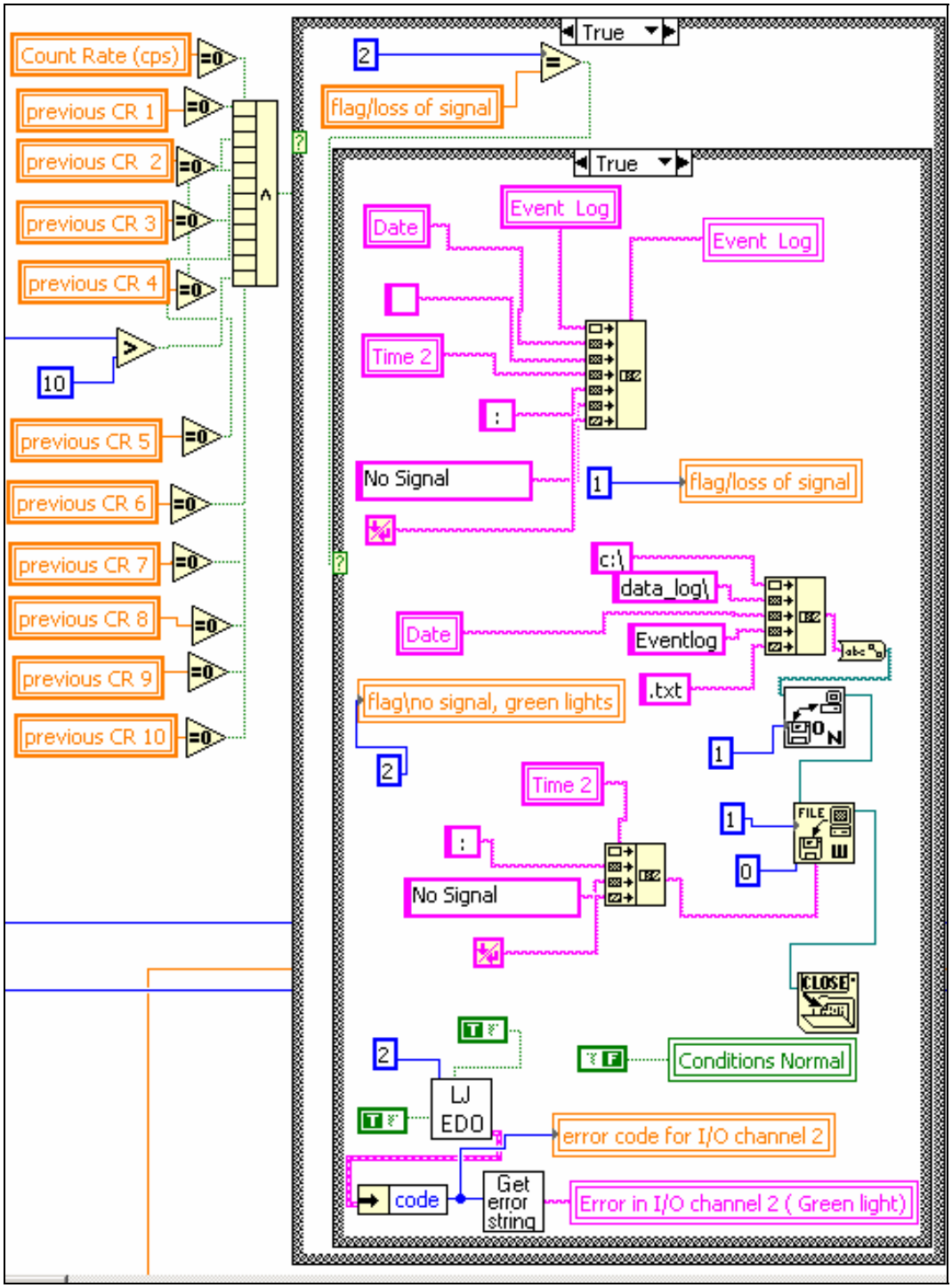


Figure 5-5: When the count-rate is continuously zero, the "NO Signal" error will be reported in the "Event log" box and saved in the "*eventlog.txt" file.

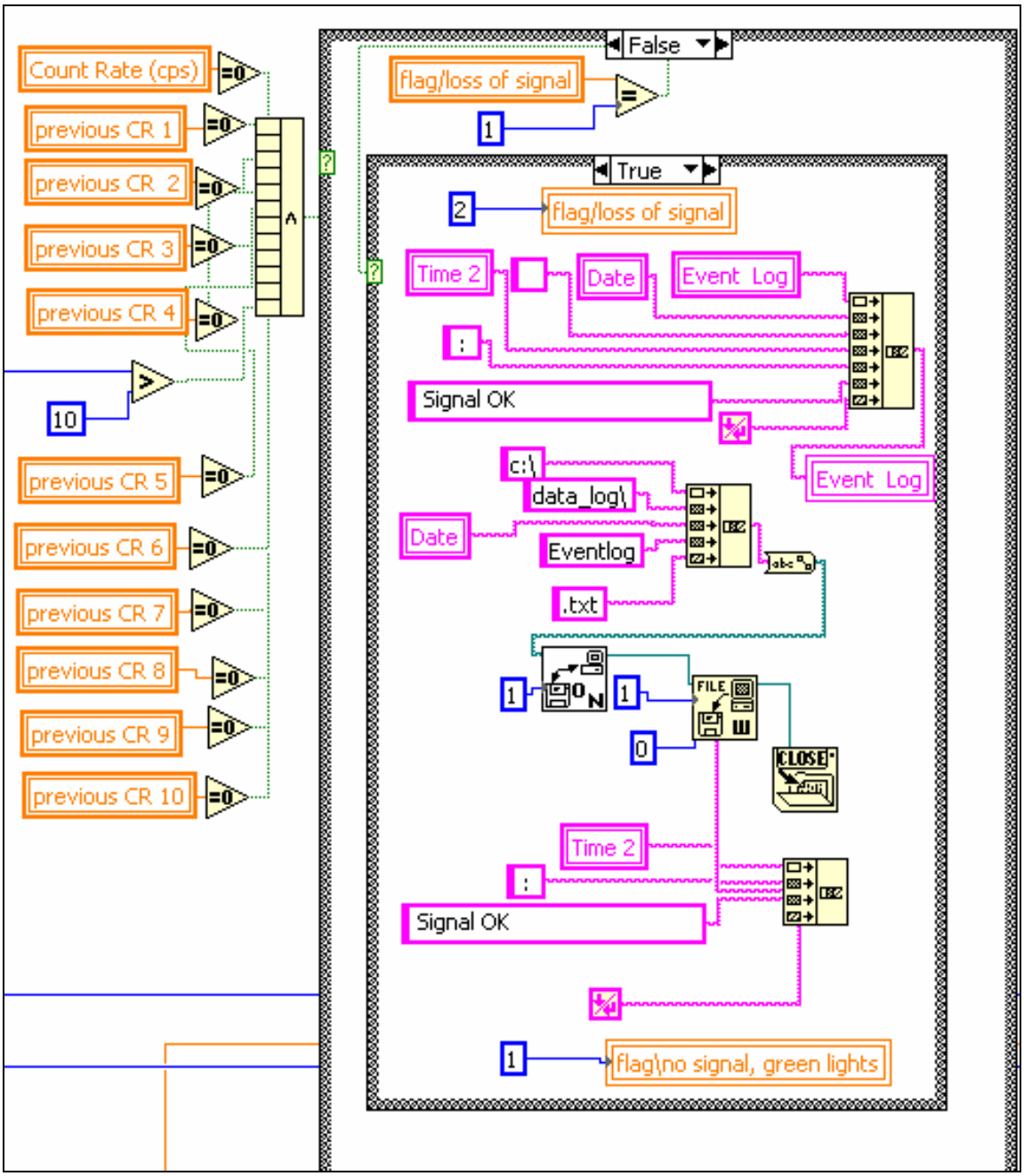


Figure 5-6: Signal reestablishment is reported in the "Event Log" box and saved in the "eventlog" file.

When the program starts receiving the signal properly, the message of "Signal Ok" will be appeared in the "Event Log" box. Figure 5-6 is the code for this part. As it is

shown in both figures the time of these messages, "No Signal" and "Signal Ok", is recorded in the "*eventlog.txt" file.

The variable of " flag/loss of signal" that is "2" for "No Signal" error and "1" for "Signal Ok" message, is set to "2" at the beginning of the execution of the program that is not shown in these figures.

As it is apparent by looking at these two pictures, Figure 5-5 and Figure 5-6, first case structure is same in both. In Figure 5-5 the "True" case and in Figure 5-6 the "False" case of this case structure is used. The "False" cases of the second case structures in both figures are empty. Note that the second case structures are completely different in two figures. The "False" cases of them are not shown there.

Figure 5-7 shows the result of these parts of the code in the "Event Log" box of the main screen in case of not receiving any signal from the counter.

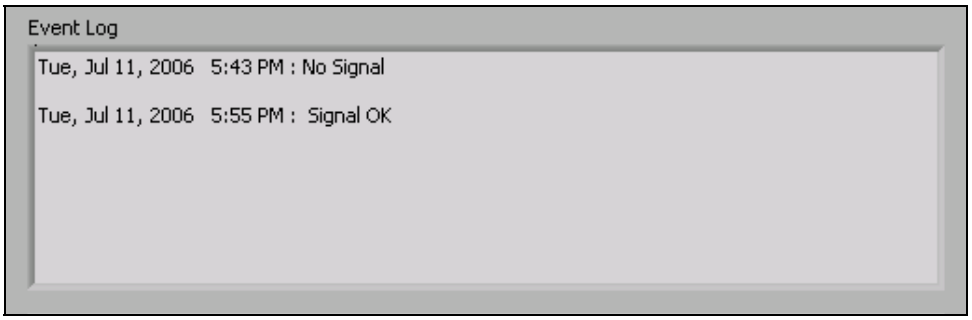


Figure 5-7: The appearance of the Event Log box in the main screen reporting two events: first the system has lost the signal, and then the signal is reestablished

As shown in Figure 5-5, "No Signal" error turns off both the flashing green light of the screen and the green light of the Signal Tower. The variable of "flag\no signal, green lights" is for setting the lights to return to normal situation. It is "2" at loss of signal condition, and "1" at normal condition. It is also set to "1" at the beginning of the program. As it will be shown in the "Alarm and Annunciation Activity" section of this chapter, one of the conditions of having the green lights "On" is the not equality of "flag\no signal, green lights" to "2".

To continue any process in the system and to have a correct count-rate, the program must not have any error in communication between CNT port of LabJack and

the PC. Any error will be reported in the "set up" page, in the section of "LabJack setting" that is shown in Figure 5-8. These errors make the green lights of the screen and the Signal Tower green light "Off".

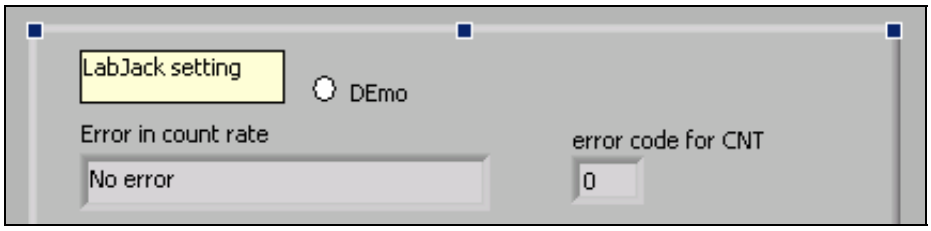


Figure 5-8: Any error in communication between CNT and PC will be reported in this section that is located in the "set up" page of the front panel.

5.2.2 Using the Soundcard to Count the Signals, An Unsuccessful Effort!

Before employing LabJack U12, we attempted to count the signals using the soundcard of the PC. Although this effort did not give us the desired result, in this section, the main functions that were used to communicate with the soundcard are discussed. During the description of these functions, it will be figured out why this effort was not successful.

The first function for this task is "SI CONFIG" that is shown in Figure 5-9. With this function, the soundcard can be configured for its input operation. The input of "sound format" is to set up the sound operation. This input has three parts. One part is the "sound quality": the sound operation is stereo or monaural? Another part specifies the "sample rate" for the sound input operation or the update rate for the output operation. The numbers that can be used are: 8000, 11025, 22050, or 44100. Another part of the "sound format" is to identify the "bits per sample". The sound operation can be set for 8 or 16 bits per sample.

The "buffer size" is another important input of the "SI CONFIG" function. The LabVIEW program uses this size of internal buffer to transfer data from a soundcard.

One of the errors that might happen when this function is used is "Overwrite" error. This error can be fixed by increasing the size of the buffer in this input.

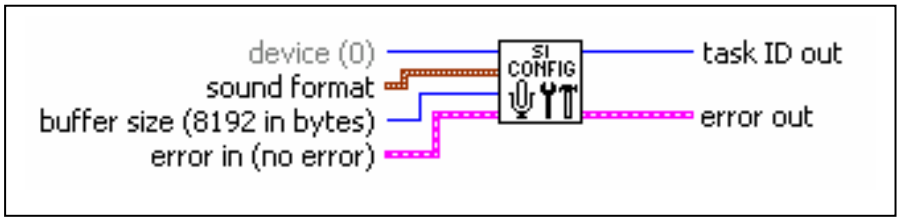


Figure 5-9 : The appearance of "SI CONFIG", the first function that is used to communicate with the soundcard of the PC.

The next function after "SI CONFIG" is "SI START". The form of this function is seen in Figure 5-10. The "task ID out", the out put of "SI CONFIG", is connected to the "task ID in", the input of "SI START". If the soundcard has not run already, this function will activate it to begin accumulating incoming data. But if the soundcard is running, this function has not any effect in the program.

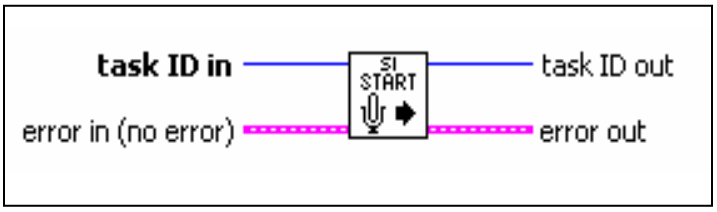


Figure 5-10: The function of "SI START".

The **"**SI READ" is the key function in the series of functions used to communicate with the soundcard. After configuring the sound and activating the soundcard, as explained above, the data is read from the input of the device using the function of **"**SI READ". Without buffering, "SI READ" waits until the new data is arrived. "Overwrite" error can accrue if for any reason the unbuffered data is over written. In this case no data will be returned by this function. Obviously, "SI READ" will return the data after buffering, if that data has arrived in the device buffer.

One of the frequent errors in the program when the code was first developed to use the soundcard for counting the signals was the "Overwrite" error. In this case, the

size of the buffer needs to be increased. But this increase makes the program too slow that results in missing some of the signals arrived at the input of the soundcard. That's why the sound card can not be reliable to count the signals.

As connection between "SI READ" and "SI START", the input of "task ID in" in the "SI READ" is wired to the out put of "task ID out" in the "SI START".

Among the different outputs of "SI READ" for different sound quality (mono or stereo) and bits per sample, the one that had been configured in the sound format of "SI CONFIG" should be chosen. This output is a very important one. It includes all the data of the sound that can be demonstrated in a waveform graph. The count-rate is the number of the peaks of this wave that is measured in another module.

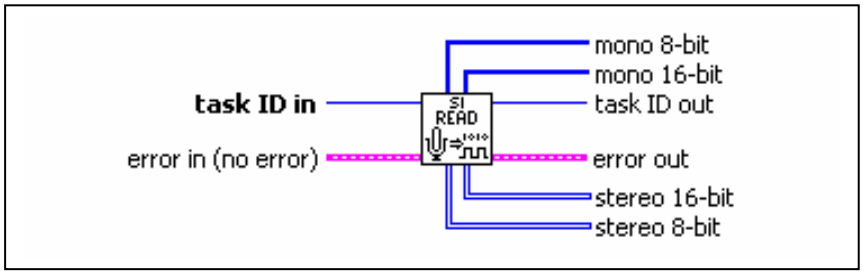


Figure 5-11: The appearance of "SI READ".

"SI CLEAR" is the last function of the SI (sound input) series. The sound input of the soundcard specified by the "task ID in" will be closed by this function. Also, all the resources that the device has been using will be released to the PC system. The reading sound is taking place within the main while loop of the code, but the function of "SI CLEAR" is located out of this loop.

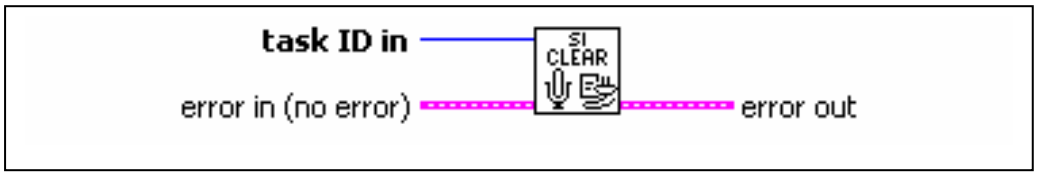


Figure 5-12: The function of "SI CLREAR" to close the sound input device.

5.3 Considering Two Modes

As previously described, the charcoal filter needs to be changed once in a while to a make the system accurate for counting the new iodine that is trapped in the filter. Apparently, when the operator is changing the filter, the program cannot calculate the Iodine Concentration nor do other normal tasks. Therefore, it can be said that there are two modes in the system: Normal Mode and Filter Change Mode.

Before start filter changing, the operator should change the statues of the toggle switch that is shown in Figure 5-13. This control switch is located in the "set up" page of the front panel .The mechanical action of this toggle is "switch when pressed". When changing filter is finished the toggle should return to the normal mode to start the initializing period. It is important to know that at the start of the execution, the program must not be in the Filter Change Mode.

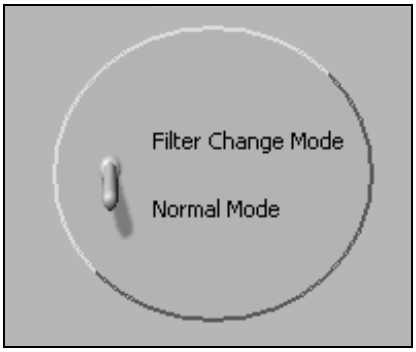


Figure 5-13: Toggle switch to adjust the mode of the system.

Since this toggle switch is not in the main screen, in order to provide a quick and easy observation of the Filter Change Mode, there will be a report in the "Event Log" box of the main screen about the beginning of the Filter Change Mode. Figure 5-14 shows the corresponding code. As it is seen in this figure, the time of the "Start Filter Change" is saved in the "Eventlog" file and also in the daily data file.

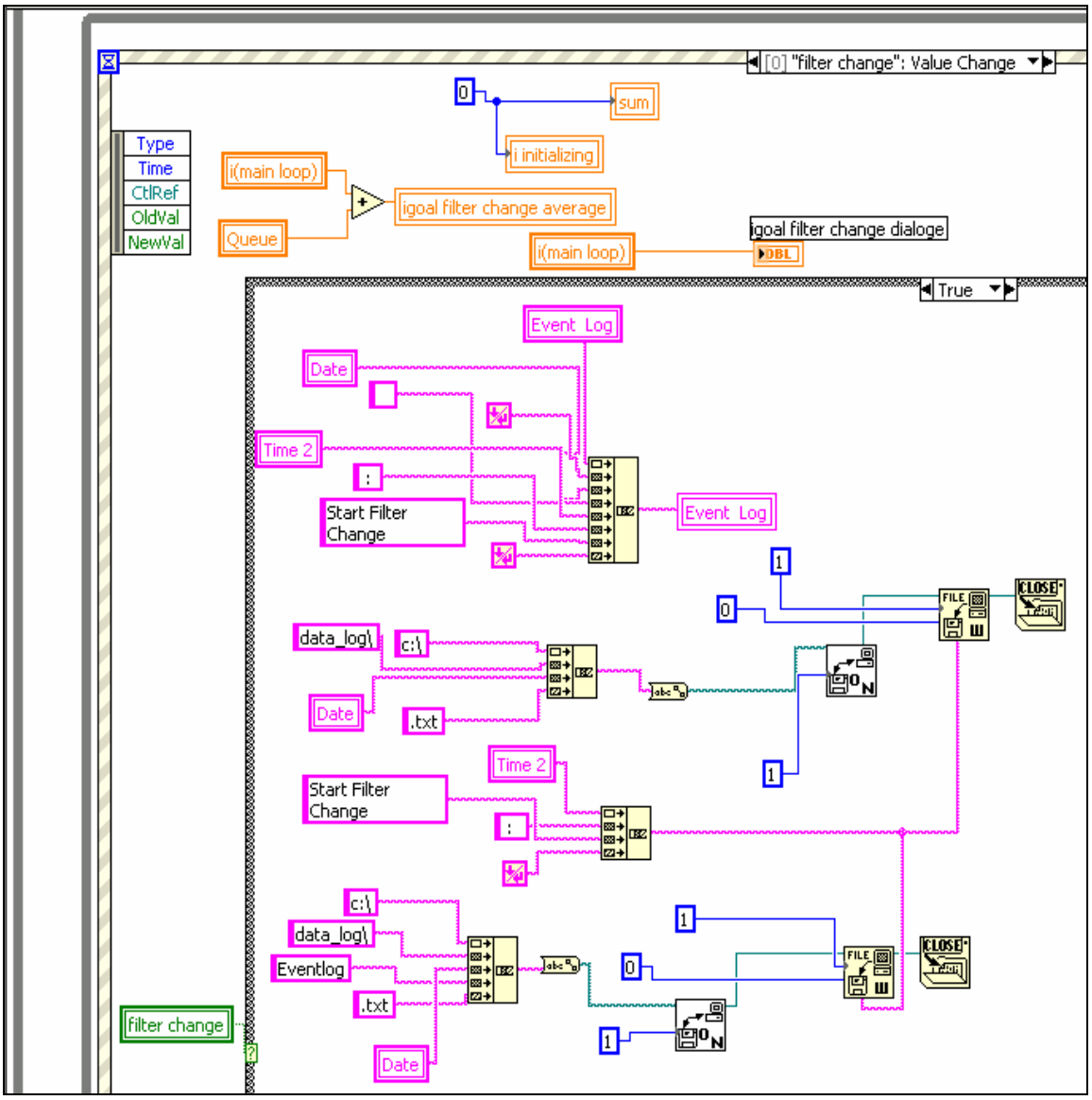


Figure 5-14: Reporting and saving the start time of the Filter Change Mode.

Figure 5-15 shows the process of reporting the exit from the Filter Change Mode in the "Event Log" box of the main screen. Like the "Start Filter Change", the time of the "Filter Change Complete" is saved in the both file of "Eventlog" and the daily data file.

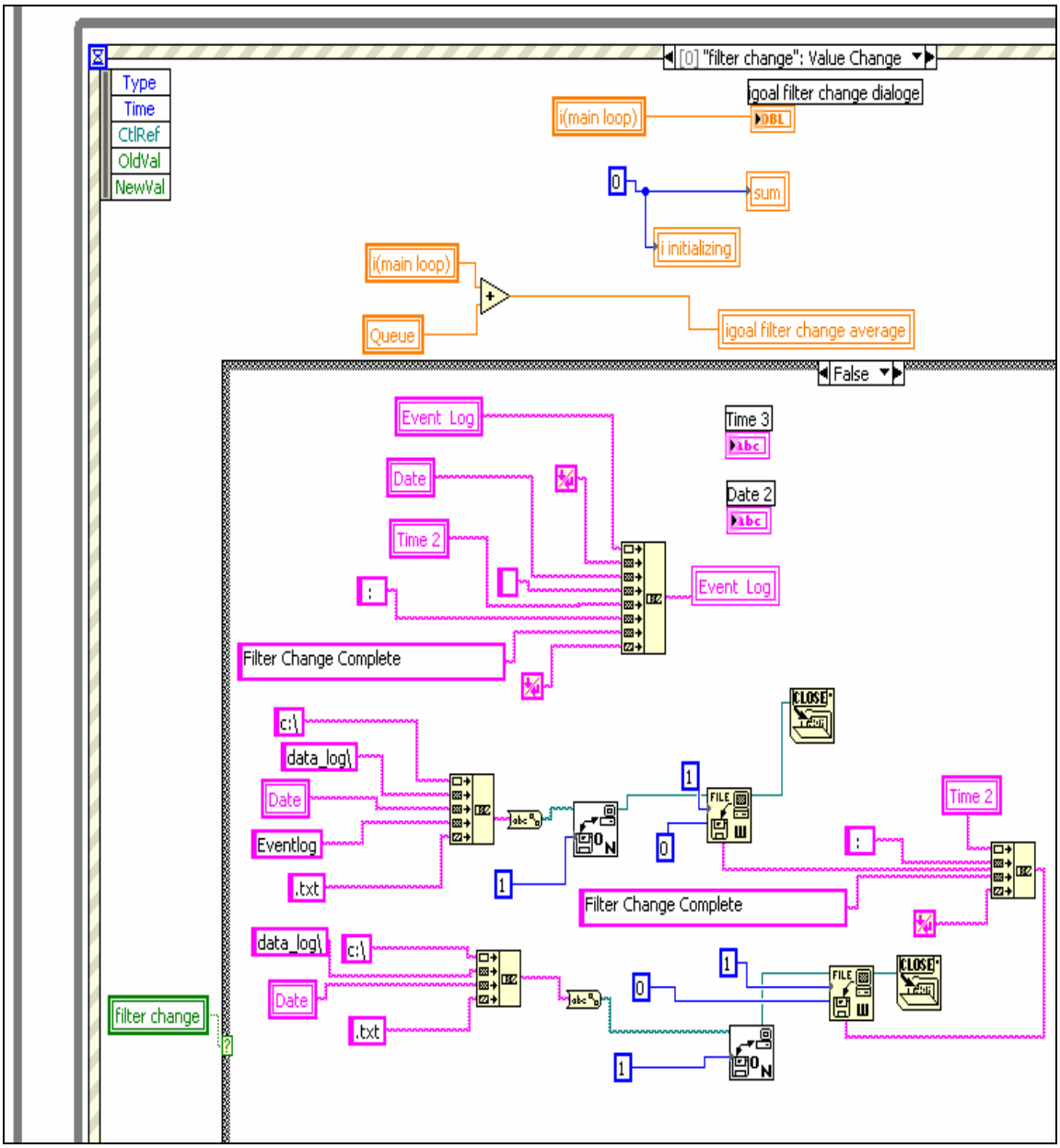


Figure 5-15: Reporting the time of "Filter Change Complete" in the "Event Log" box and saving it in the daily data file and the "Eventlog" file.

The description of arriving into and exiting from the Filter Change Mode stays in the "Event Log" box until the end of the day. Figure 5-16 shows the "Event Log" box with such information in it.

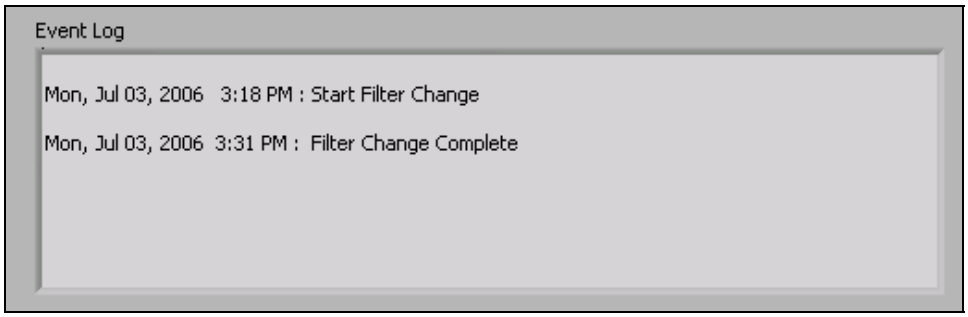


Figure 5-16: "Event Log" box of the main screen contains the information of starting and completing the filter change.

After entering into the Filter Change Mode, the green light of the Signal Tower and also the flashing green light of the main screen will be "Off". The corresponding code for this part is described in the "Alarm and Annunciation Activity" section of this chapter. The staff can be informed of the reason that the green lights are "Off" by glancing at the "Event Log" box in the main screen.

It is obvious that the program can not be in the initializing period in the Filter Change Mode because of its disability in accumulating accurate data at this time. But in case the mode switches to Filter Change Mode during the initializing period, the flashing indicator of the initializing phase in the main screen needs to be deactivated. Figure 5-17 shows the process of making this indicator inactive at the Filter Change Mode. Then after changing the filter, the initializing period will be started from the beginning.

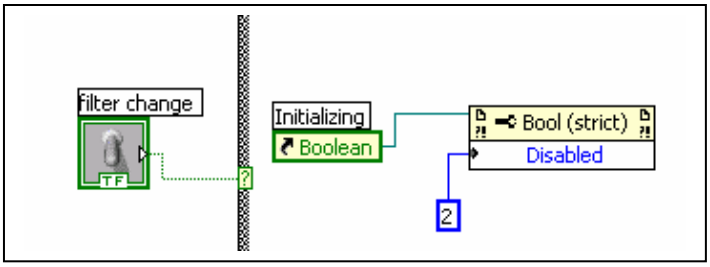


Figure 5-17: In the Filter Change Mode the indicator of initializing phase is inactive. The case structure is in the "True" case at this figure.

By entering into the Filter Change Mode, the average amount of Iodine Concentration of the old filter, i.e. running average of that moment, is transferred to a graph in the main screen. This graph is called “Long Term Average I-125 Concentration (Bq/m^3)”. As the filter is usually changed weekly, this amount is recorded as weekly average or long term average. The code of this task is widely described and shown in the “Plotting the Graphs” section of this chapter.

Since at the Filter Change Mode we have no estimation for Iodine Concentration, entering to or staying in this mode should not happen by mistake. Therefore, to increase the safety margin, a dialog box is set to pop up at the beginning of the Filter Change Mode to show arriving to this mode. This dialog box also asks to return to Normal Mode after changing the filter.

Figure 5-18 shows the corresponding code. The “One button dialog” function is used here. This dialog box contains a message and a single button. The picture of these dialog boxes are shown in Figure 5-19 and Figure 5-20 .

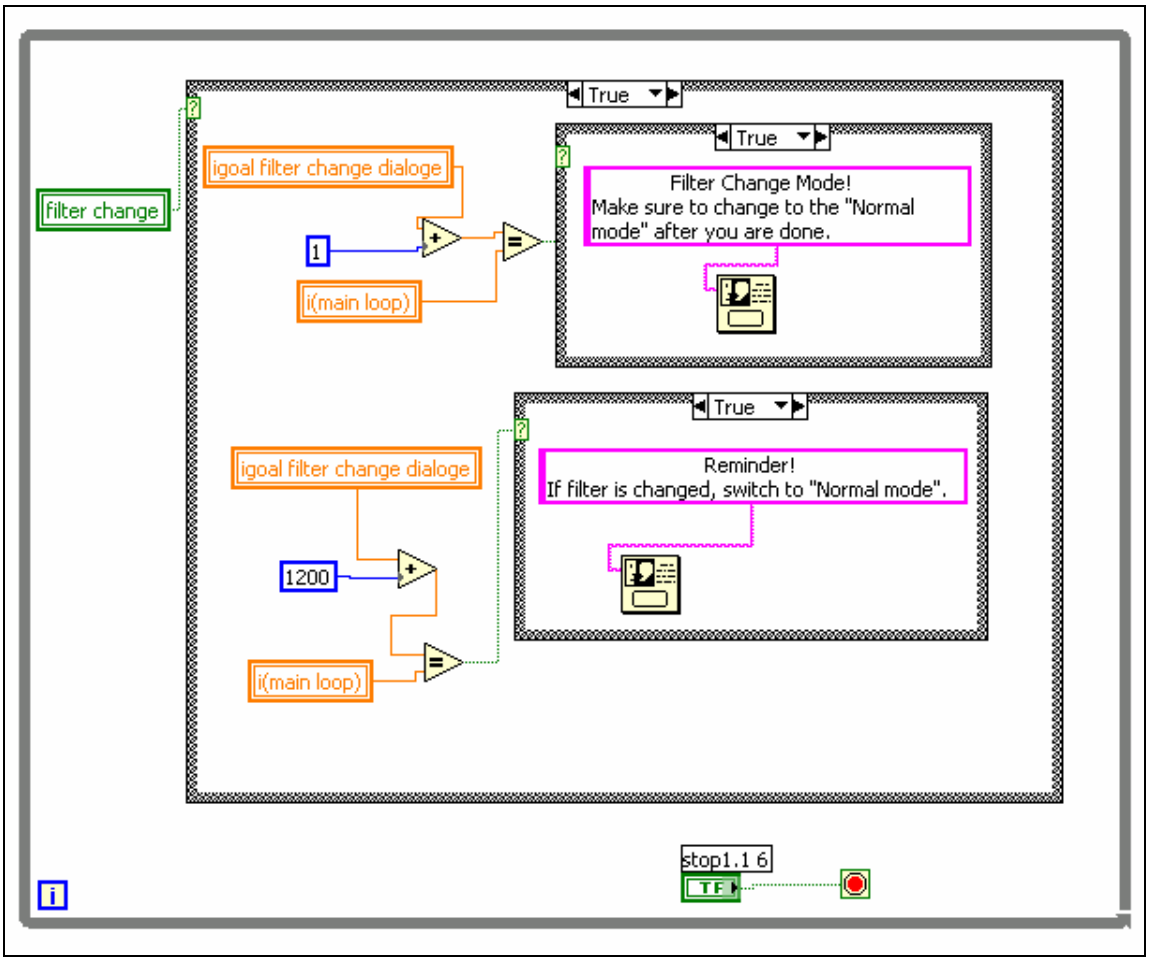


Figure 5-18: The code of the dialog boxes in the separate while loop.

The second dialog box is in fact a reminder for the operator to make sure to return to the Normal Mode after s/he has changed the filter. As it is seen at the code of Figure 5-18, this dialog box will pop up after twenty minutes of entering to the Filter Change Mode. But if the mode has been already changed by then, the dialog box does not show up. Since filter changing usually takes around fifteen minutes, if returning to the normal Mode is forgotten after twenty minutes, this reminder can be very helpful. These boxes will not go away until the “OK” buttons are pushed.

Figure 5-18 shows that the whole process should be in a separate while loop, not in the main loop and not in common with any other procedure. Otherwise, the program is paused until the operator pushes the "OK" button.

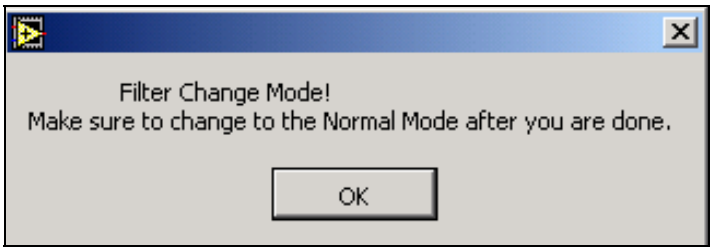


Figure 5-19: This dialog box appears just after entering to the Filter Change Mode.

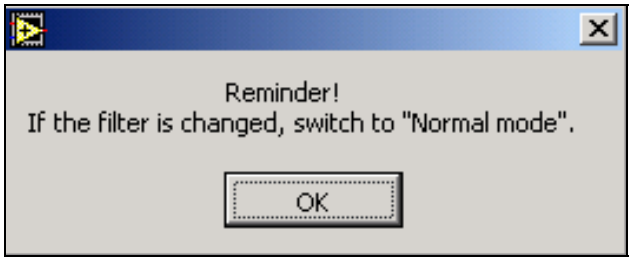


Figure 5-20: It appears twenty minutes after arriving to the Filter Change Mode if the program still stays in this mode.

5.4 Calculating the Iodine Concentration and Estimating the Error

According to the previous analysis, Iodine Concentration depends on the "Slope of Count-Rate in Time ", "Detector Efficiency" and "Pumping Flow Rate" according to:

$$Iodine\ Concentration = 6\times10^{6}\ \frac{Slope\ of\ \ Count\_Rate\ in\ Time}{Detector\ \ Efficiency\times Pumping\ \ Flow\ Rate}$$

Equation 18

This equation is used in the main code as shown in Figure 5-21. The indicator of "IC main" in the figure shows the calculation result. However, as seen in Figure 5-23, the

amount of "IC main" is not always equal to the amount of Iodine Concentration that is demonstrated in the front panel. "IC main" indicates the estimation of Iodine Concentration only after the initializing period in the Normal Mode.

As it was mentioned before, the "Least Square Solution" method is employed to find the best estimate for the "Slope of the Count-Rate in Time".

The equation for the "Slope" based on the "Least Square Solution" was derived in chapter 2. The result is shown below:

$$m = -\frac{6}{N(N+1)}\sum_{n=0}^{N-1} y(n) + \frac{12}{N(N^2-1)}\sum_{n=0}^{N-1} n y(n) \qquad \textbf{Equation 19}$$

N is the number of samples and $n = 0,1,...,N-1$. In MASIS $y(n)$ which is the count rate counted by the LabJack.

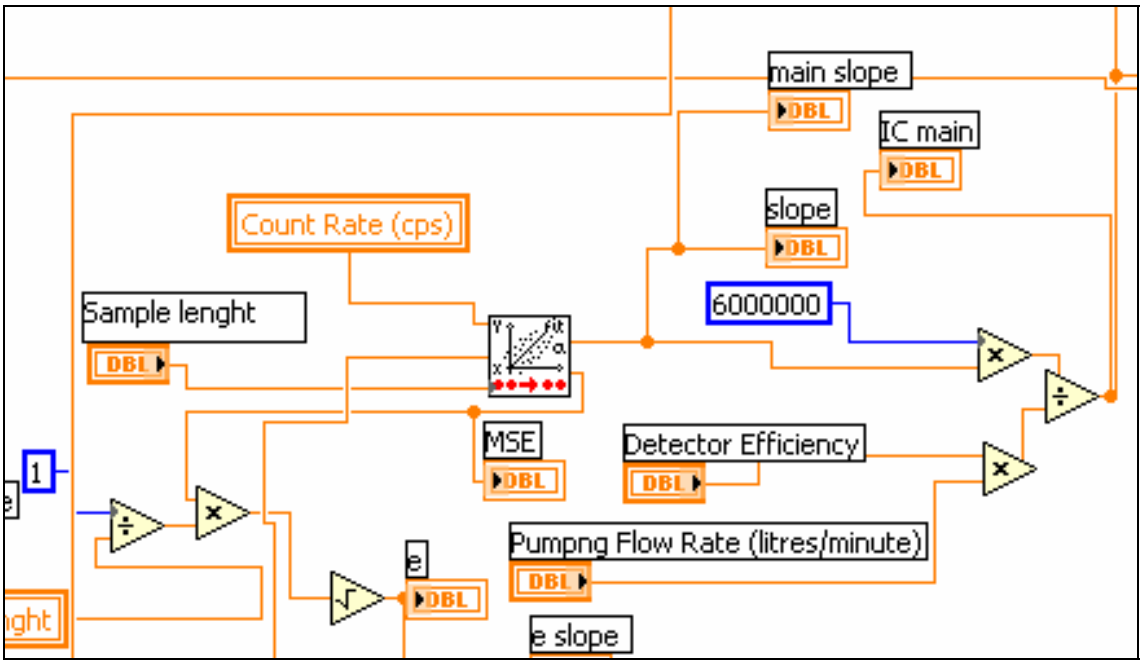


Figure 5-21: The part of the code that shows how Iodine Concentration is calculated. More detailed description of this figure is given in Appendix6

There are several functions in LabVIEW that meet our need to find the slope from "Least Square Solution" in the code. The simplest one is the "Linear Fit Coefficients PtByPt". But as seen in Figure 5-21, this function is not used in the code. Instead, the function of "Linear Fit PtByPt "is preferred. Its appearance is shown in Figure 5-22.

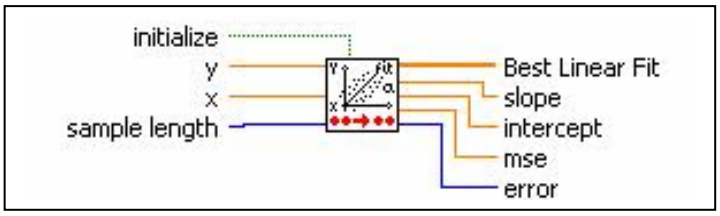


Figure 5-22: The function of "Linear Fit PtByPt".

This function is able to calculate the "Mean Square Error" of the estimation as well. "Mean Square Error" is one of the parameters used in the calculation of Iodine Concentration error. By applying the function of "Linear Fit PtByPt", the program does not need to employ a separate function to calculate the "Mean Square Error". In Appendix 6, it is described how these two functions, "Linear Fit Coefficients PtByPt" and "Linear Fit PtByPt", are related to each other. Because of the importance of the sub-VI of "Linear Fit PtByPt" in the main program, the code of this function is also shown there.

"Running Average" is another important parameter calculated by the program. It is the average of the Iodine Concentration from the last filter change until the current time. Obviously, the time of the initializing period after filter changing is ignored in the estimation of "Running Average". This parameter has an indicator box in the front panel and the operator can see the value of "Running average" from the main screen of the program. It is updated every second. Figure 5-23 shows the main part of the calculation of "Running average".

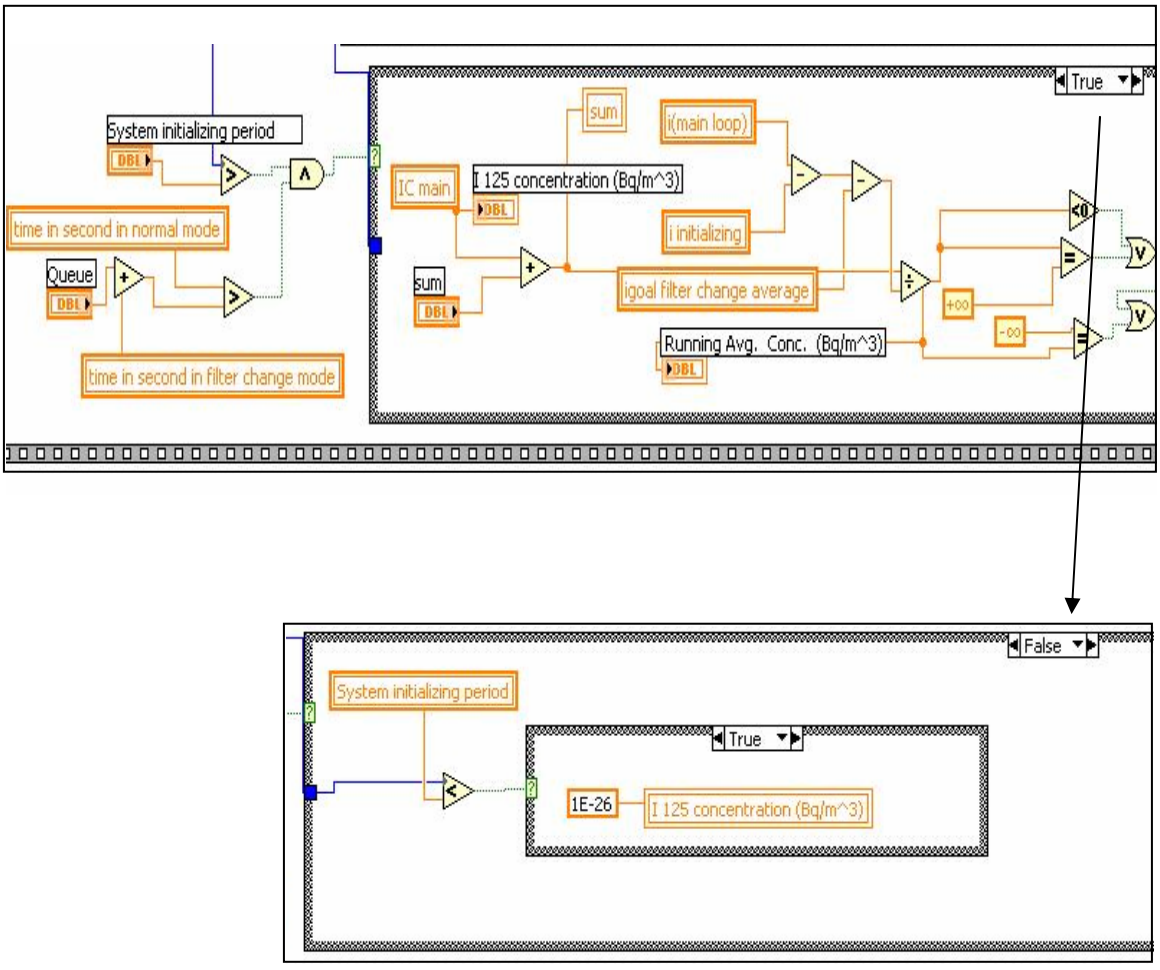


Figure 5-23: This part of the main code shows when "IC main" calculated in Figure 5-21 is equal to "I-125 Concentration (Bq/m^3)". Also this figure shows how the "Running average" of Iodine Concentration is calculated.

The error in Iodine Concentration is calculated at each time step. As previously developed, "Percentage of the Error" of the Iodine Concentration is calculated from the following equation:

$$Percentage\ of\ the\ Error = \pm \frac{Iodine\ Concentration_{(real)} - Iodine\ Concentration_{(estimated)}}{Iodine\ Concentration_{(estimated)}} \times 100 =$$

$$\pm K \frac{\frac{\Delta Error}{\Delta t}}{Iodine\ Concntration_{(estimated)}} \times 100$$

Equation 20

Where:

$$K = \frac{6 \times 10^6}{Detector\ Efficiency \times Pumping\ Flow\ Rate}$$

And

$$\text{Error} = \sqrt{\frac{1}{n} MSE} \qquad , \qquad \text{MSE} = \frac{1}{n} \sum_{i=1}^{n-1} (Yi - y_i)^2$$

The code for the above calculation is shown in Figure 5-24. The variable of "Previous Error" in this figure, is used to find "$\Delta Error$" in Equation 20 . "Previous Error" is the amount of "Error" in the previous iteration that is simply found by a shift register which is not shown in Figure 5-24.

In the code, "n" is the "sample length" in function of "Linear Fit PtByPt" (Figure 5-22)**.**

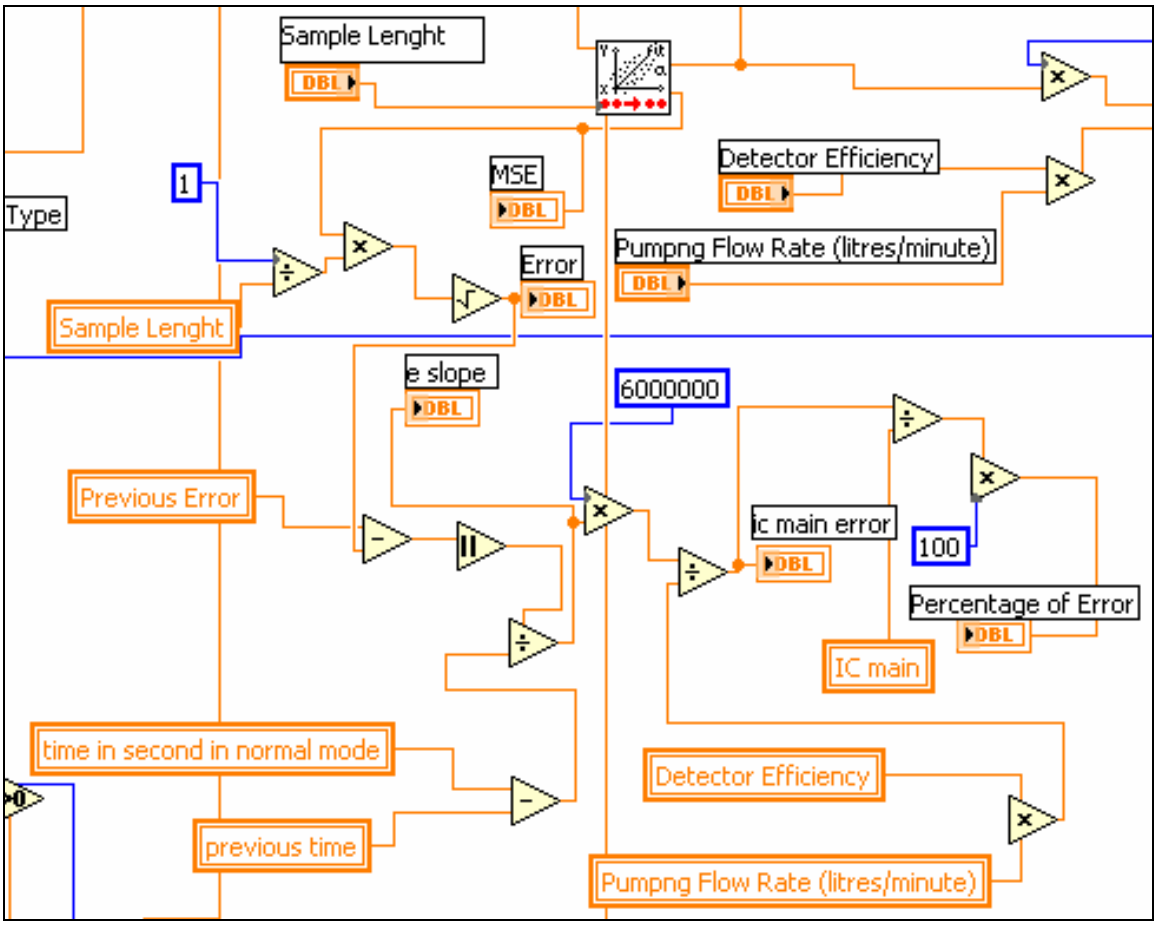


Figure 5-24: The Percentage of Error is calculated in this part of the code.

As explained in the previous chapter, all the results of the Iodine Concentration calculation and the error estimation will be functional only after the initializing period. In the initializing period, which is the specified waiting time after starting the program and also after filter changing, the amount of Iodine Concentration and the corresponding information is not accurate. Therefore, this information is not shown in the relevant indicators on the screen and is not saved in the daily data file. Instead, the starting and ending time of the initializing period is reported on the main screen and is saved in the file. Figure 5-25 shows the code of this step. The first case structure is in the "False" case. Because of the space limitation, it is not possible to show the "False" mark in this

case structure at the figure. The variable of "save\flag", which is used in the code of Figure 5-25, is set to zero at the start of the execution of the program.

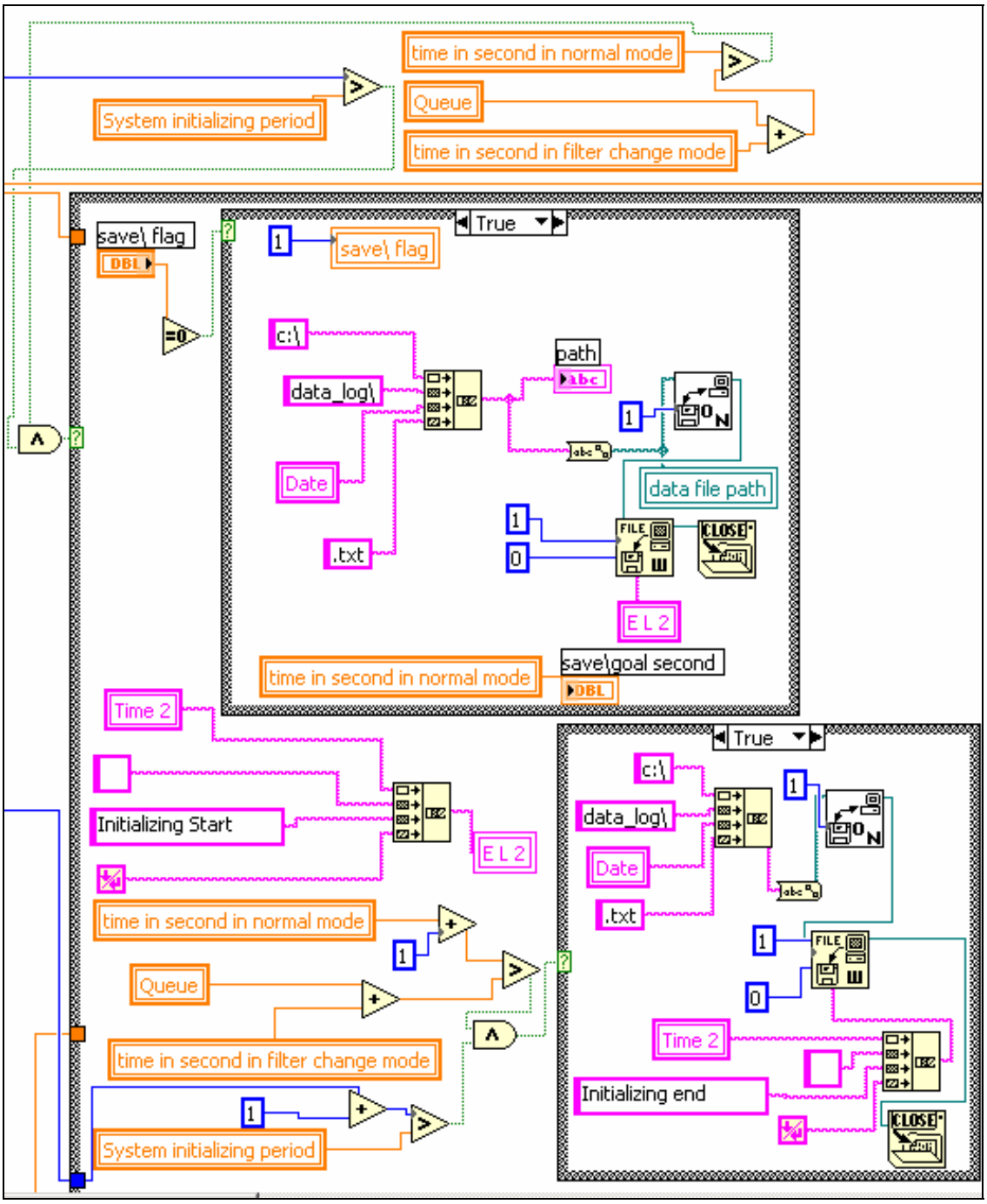


Figure 5-25: The part of the code for reporting of arriving into and exiting from the initializing phase.

Also in the initializing period, the green light of the Signal Tower and the screen green light are "Off ". The code of this procedure will be presented in the Alarm and Annunciation section of this chapter.

A flashing indicator, shown in Figure 5-26, signifies the initializing period in the main screen. It is activated in the initializing period, so by looking at the main screen the staff members quickly understand why the green light is "Off".

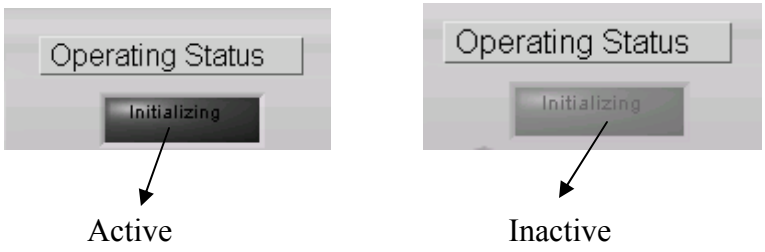


Figure 5-26: The initializing indicator in the main screen. It flashes, when it is active and it is disabled and grayed out in the inactive duration.

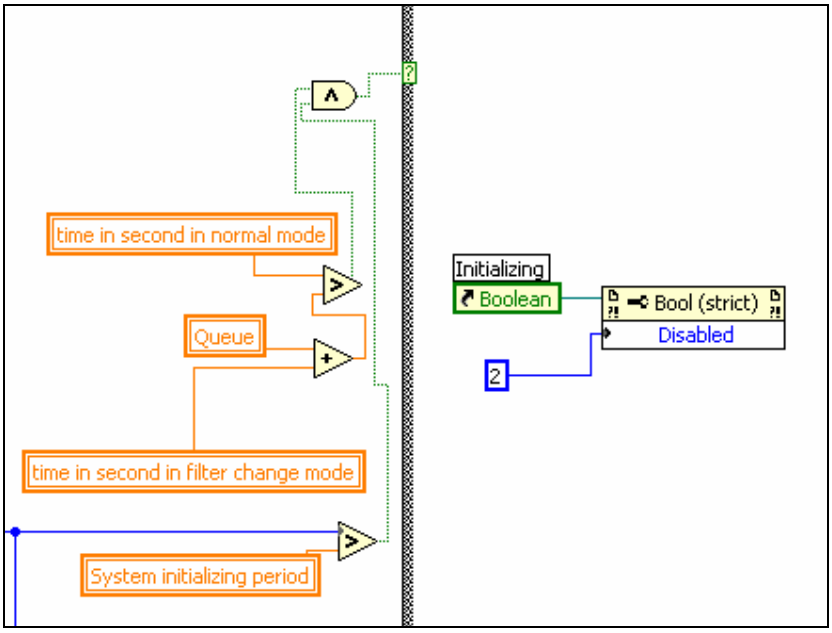


Figure 5-28: The indicator is disabled and grayed out outside the initializing period.

Figure 5-27 and Figure 5-28 are parts of the code that control the initializing indicator in the Normal Mode. Figure 5-27 shows how this indicator flashes in the initializing period. The "0" or the "2" wired to the "disabled" property enables or disables (and grays out) the initializing indicator respectively.

Figure 5-28 is part of the code to deactivate the initializing indicator when the program is out of the initializing period.

The first case structure in Figure 5-28 and Figure 5-27 is the same but the opposite case. In Figure 5-27 the case structure is in the "False" case and in Figure 5-28 it is in the "True" case.

Since the program stays in an infinite loop in the Filter Change Mode, it never reaches to the step that is shown in Figure 5-28, and hence, does not deactivate the initializing indicator. In order to make sure that this indicator is inactive when the program is in the Filter Change Mode, the program disables and grays it out upon arriving at this mode. If the Indicator has already deactivated, this task does not have any effect on the program. The code of this step will be shown in the section of "Considering Two Modes" of this chapter.

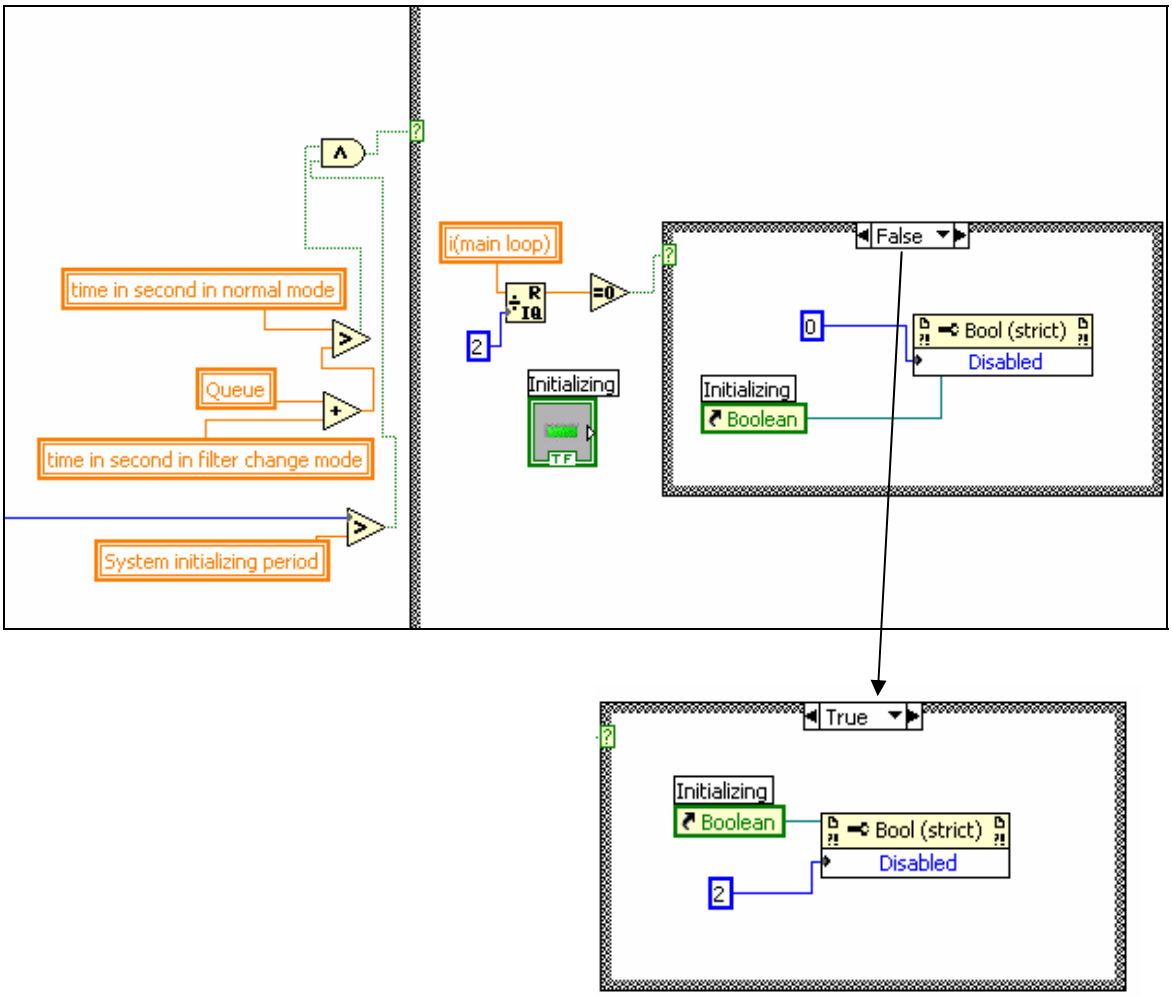


Figure 5-27: When the program is in the initializing period, the initializing indicator flashes.

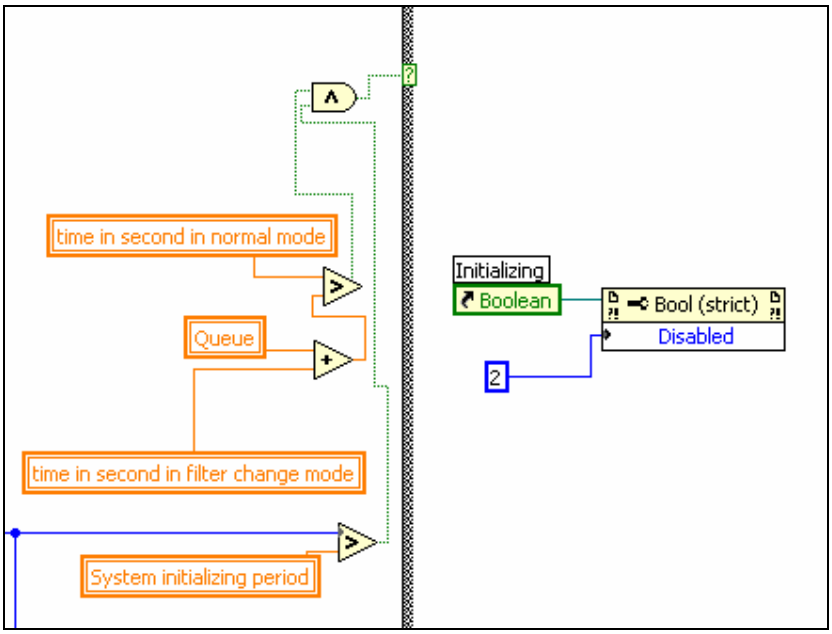


Figure 5-28: The indicator is disabled and grayed out outside the initializing period.

5.5 Alarms and Annunciation Activity

Alarm and Annunciation system of MASIS includes a Signal Tower mounted on the wall, flashing lights on the PC screen, and auto-sending of alarm email messages to the reactor crew. A detailed description of this system is presented in the following sub-sections.

5.5.1 Controlling the Signal Tower

Signal Tower is the main part of the alarm and annunciation system. It contains three lights (green, amber and red) and a buzzer.

Green light indicates the normal situation. i.e. everything is alright with the system, Iodine Concentration is accurate and its amount is within the safe limit. The conditions that make the green light "Off" are listed as below:

- Filter Change Mode
- Initializing phase that is the period of time that the program is busy for collecting data.

- Loss of signal in the system
- Error in any of the four I/O channels of LabJack connected to the Signal Tower
- Error in the count-rate and the CNT input connection of the LabJack
- Unsafe range of Iodine Concentration that can make the amber light or the red light and the buzzer "On"

Each of these conditions has an indicator on the screen that helps the operator to find out the reason that the green light is "Off". For example: "LabJack Setting" section in the "Set up" page shows the exact explanation of any error related to the LabJack. Also the Filter Change Mode, Initializing period, and some errors in the system are reported on the "Event log" box.

Parts of the main code that describes the control procedure of the green light are shown in Figure 5-29. This figure shows the situations that make this light "On" or "Off".

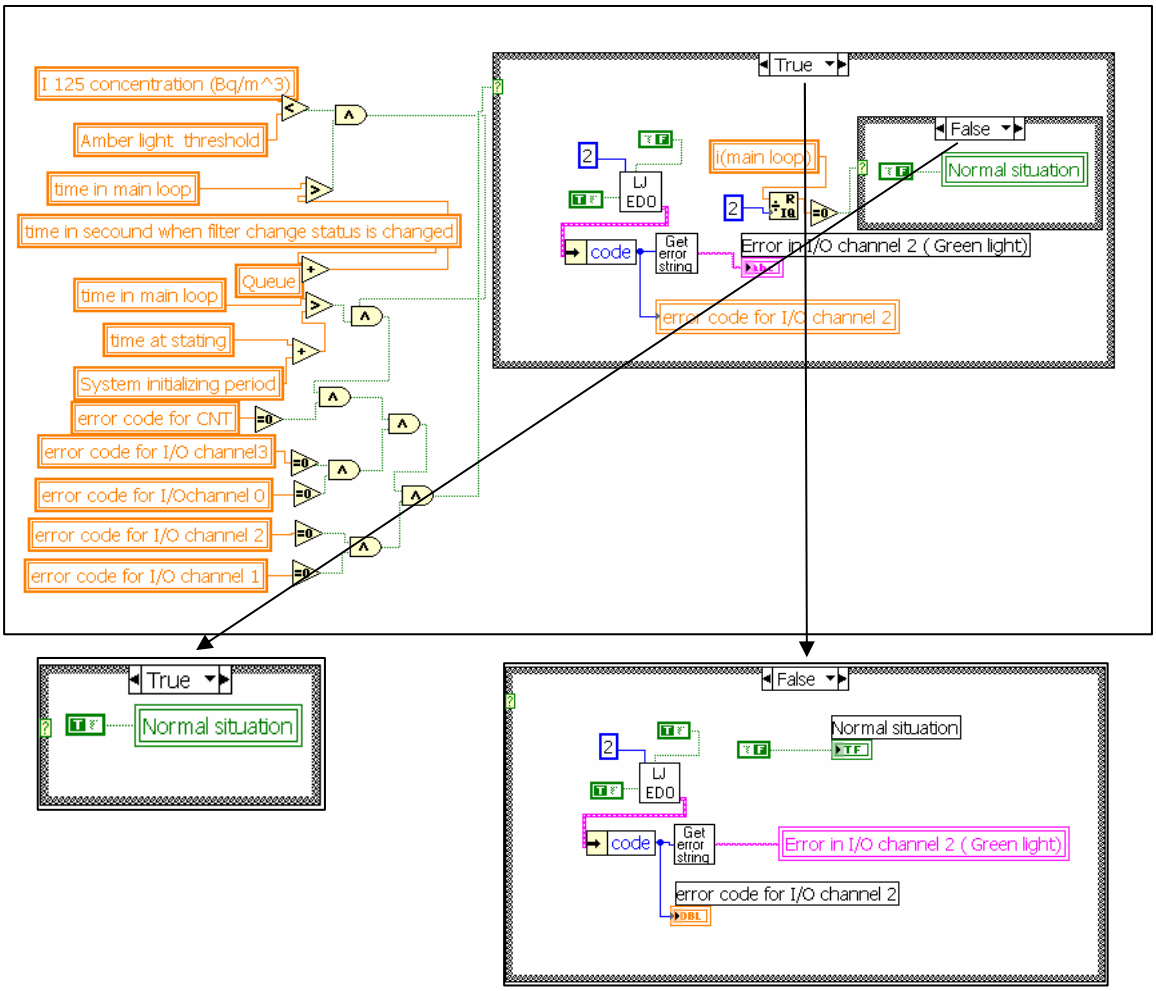


Figure 5-29: Controlling the Signal Tower green light and the green light of the main screen.

Amber light indicates "Warning" situation that means "Iodine Release Occurring". The threshold of the Iodine Concentration for amber light varies depending on the locations and is adjustable for the operator through the password protected area of the code.

Figure 5-30 shows how the amber light is controlled.

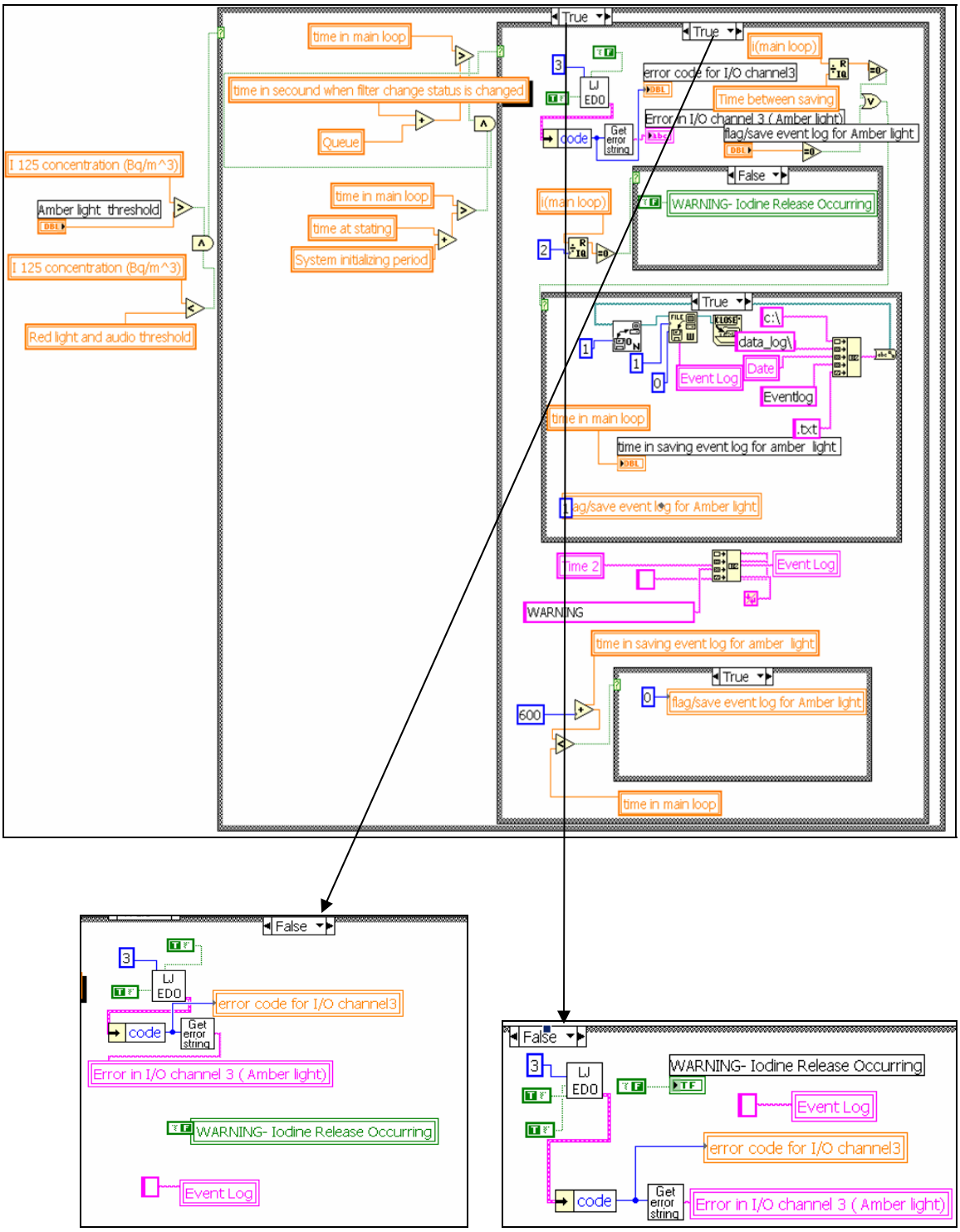


Figure 5-30: This figure shows how the program controls the signal tower amber light and the amber light of the screen.

Red light turns on at higher amount of Iodine Concentration, above the maximum threshold for the amber light. Flashing red light is the "Alarm" that indicates the approach to the administrative release limit. Figure 5-31 shows parts of the code to manage the red light.

There is also a buzzer for this limit. To stop disturbing the personnel, the operator is able to suspend the audio alarm system from the main screen after recognizing the alarm. In the case of an ongoing alarm situation, it will be automatically reactivated after ten minutes. This time is also adjustable through the password protected area. Figure 5-32 is part of the main code associated with the audio alarm.

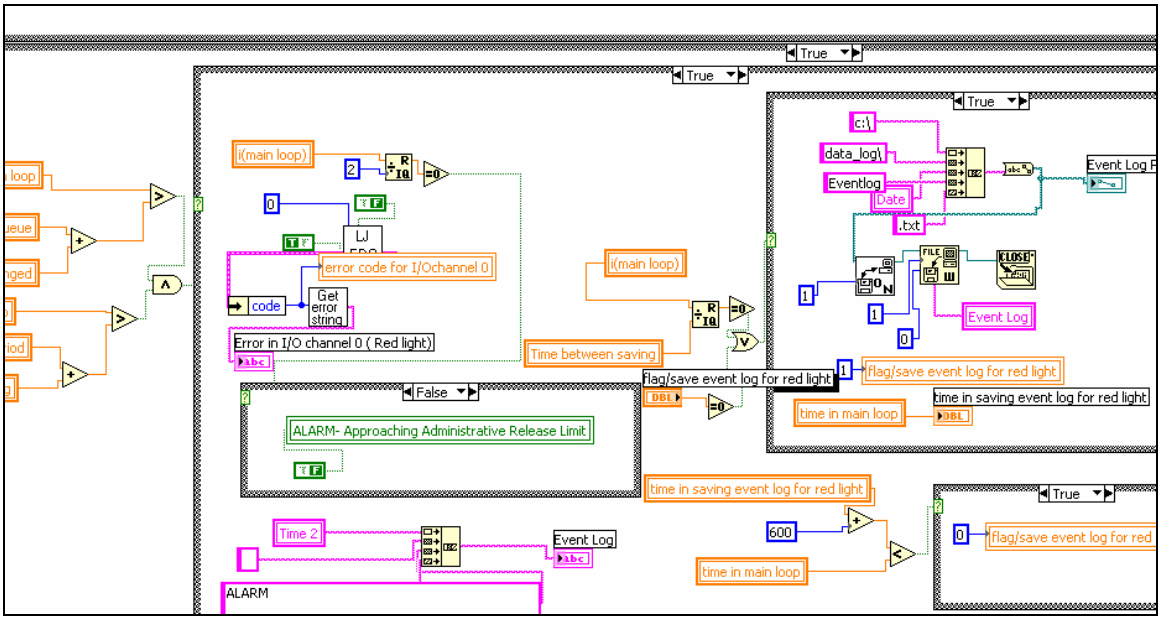


Figure 5-31: This part of the code corresponds to the flashing red light of the signal tower and the screen.

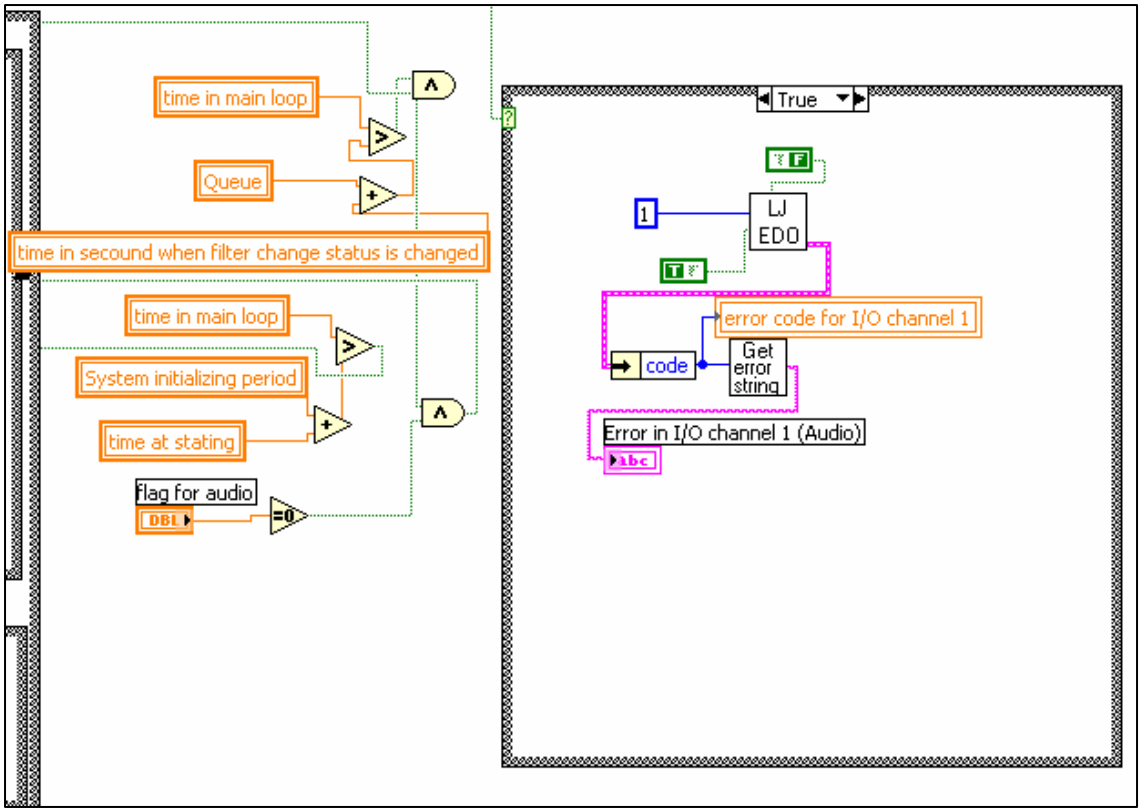


Figure 5-32: Controlling the buzzer of the Signal Tower.

The important common function among the codes of Figure 5-29, Figure 5-30, Figure 5-31 and Figure 5-32 for controlling the lights and the buzzer is "EDigitalOut". The appearance of this function is shown in Figure 5-33.

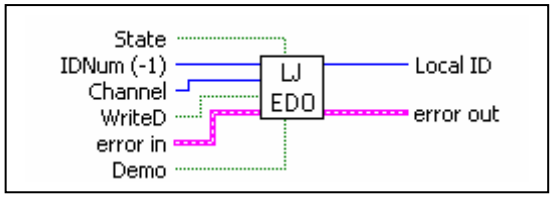


Figure 5-33: The function of "EDigitalOut".

As previously mentioned, to send a signal from LabJack U12 to Signal Tower, four digital I/O channels (for three lights and a buzzer) from DB25 connector of the

LabJack are used. To communicate with these channels as outputs, the function of "EDigitalOut" is employed. It takes usually 16 milliseconds for this function to execute in windows. However, the execution can take up to 20 milliseconds and the maximum update rate is about 50 Hz for each channel.

This function is able to set or clear the state of a particular channel number out of DB25 pins or screw terminals. As shown in Figure 5-29 to Figure 5-32 this task is performed by connecting "True" or "False" constant to the input of "state" of the "EDigitalOut" function and configuring the desired channel. It should be noticed that since the output signals of these four channels are digital, their status, which is either zero or one, remains untouched until the next output signal updates the status.

"EDigitalOut" is an easy function and a simplified version of the lower level function "DigitalIO", which works on all 20 digital pins. In first loading of DLL (ljackuw), without knowing the direction and state of the lines, it assumes output states are low and all directions are input. While the DLL (ljackuw) is keeping track of the current direction and output state of all lines, this easy function can manage a particular channel without affecting the other lines.

Appendix 7 shows the code of sub-VI "EDigitalOut". Also more detail about the parameter description of it is available in the appendix.

5.5.2 Flashing Lights of the Main Screen

Besides the Signals Tower mounted on the wall, three lights are also displayed on the PC screen for the "Normal", "Warning" or "Alarm" condition. They have same colors as the lights of the Signal Tower and mimic their behavior. The only difference between these lights and those of the Signal Tower is that the former ones flash to provide a better vision on the screen. In case of any problem with the Signal Tower lights, these lights can take their role. Figure 5-34 shows the appearance of these lights. Figure 5-29, Figure 5-30 and Figure 5-31 include parts of the MASIS code for controlling these lights.

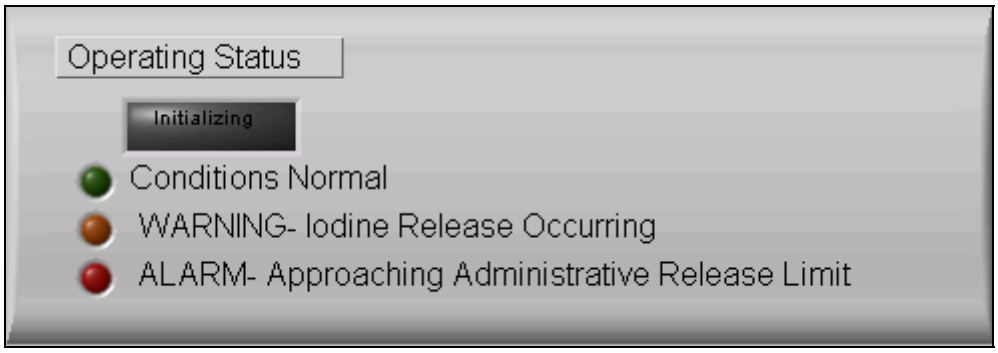


Figure 5-34: Flashing lights on the main screen.

5.5.3 Auto-sending alarm email messages

One of the features of the program for the Alarm and Annunciation System is informing the plant crew of the emergency events by auto-sending of email messages.

"SMTP Email Send File" is the main function that is employed for this part. It is shown in Figure 5-35.

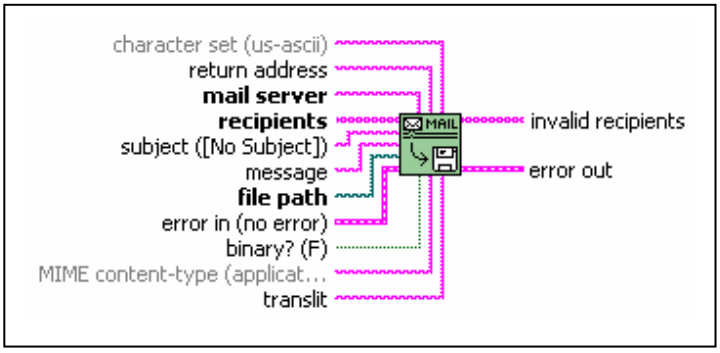


Figure 5-35: The function of "SMTP Email Send File"

LabVIEW has various functions for sending emails. The function used in MASIS, "SMTP Email Send File", has the capability to send an attachment with the email. When a warning message is sent out, the data file of the day is also attached to the email. Thus,

the operator and staff members, who are not in the site at the moment, will have more detailed information of the situation.

Another ability of this function is that the email can be sent to as many recipients as desired. It also gives the list of invalid recipients.

Figure 5-36 shows how this function is used in the program. The threshold for sending an alarm message is the same as the one for the red light and the buzzer. The threshold varies with the location where the Iodine Concentration is measured.

As shown in Figure 5-36 the first email is sent just after reaching the alarm situation. If this situation continues within the next hour, the second alarm message will be sent out. This process will repeat.

Figure 5-37 shows mail server, recipients, email subject and message can be changed by the operator. This part of the front panel is located in the password protected area and is not accessible by others.

The function of "Error Handler" is very necessary for this part of the code. Without "Error Handler", if the internet is disconnected or the mail server does not work during the alarm condition, the program encounters some interruption by the dialog boxes. The operator has to confirm these dialog boxes popping up at each iteration as they pause the program. A "Simple Error Handler" is employed to avoid any interruption in the program in case of problems with the internet.

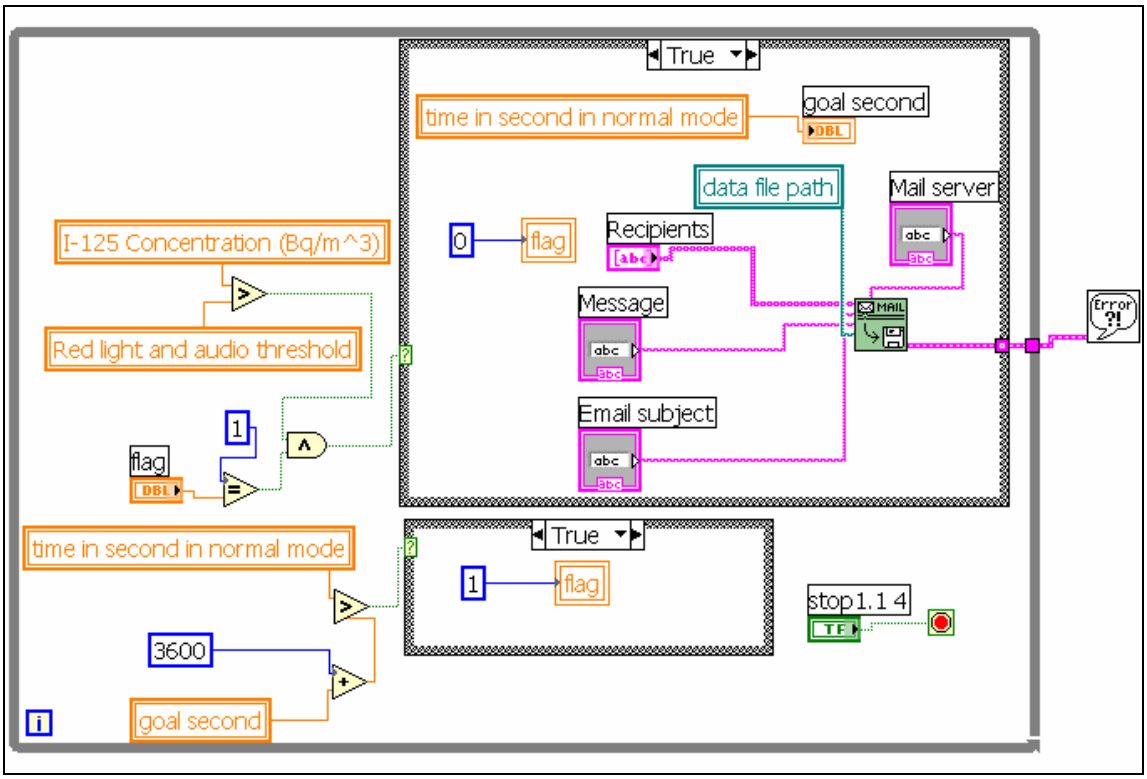


Figure 5-36: Part of the code responsible for sending email to the crew of the reactor at the time of the alarm situation. The daily data file is attached to this email. The frequency of sending the email is one hour.

Figure 5-37: Part of the Front Panel related to the auto-sending alarm email message. It is located in the password protected area.

5.6 Creating Files, Writing into and Reading from Them

One of the essential tasks of the program is saving the output data. These records are valuable for the future data analysis. Also reading from files is important for reloading data into a variable or a graph. In different parts of the code explained before, we encountered recording data into a file. This was mainly done for two purposes:

1) To save and keep data in a file for future analysis
2) To reload and read the data into the desired place when it is needed

This section is dedicated to these parts of the MASIS code.

5.6.1 Creating and Writing

There are four major groups of files that are created in the program. Daily data files, Evenlog files, Logbook files and one group of scattered files in the "variables" folder. In the following sub-sections, each of these groups will be explained separately.

5.6.1.1 Daily Data Files

A main group of the files in the program that provide important information for future data analysis are daily data files. A new text file is created every midnight in "Local Disk(C:)" and contains the data collected during the day**.** These data files are all saved in a folder called "data-log". The name of each file contains the date of its creation**.**

Figure 5-38 shows the process of creating this file and writing the data into it. The three functions are used for creating a new file or opening an existing file and writing data into it. Although there are some other functions available in LabVIEW, usually these functions are used in different parts of the MASIS code when writing data into a file is needed. Figure 5-39, Figure 5-40 and Figure 5-41 show these three functions.

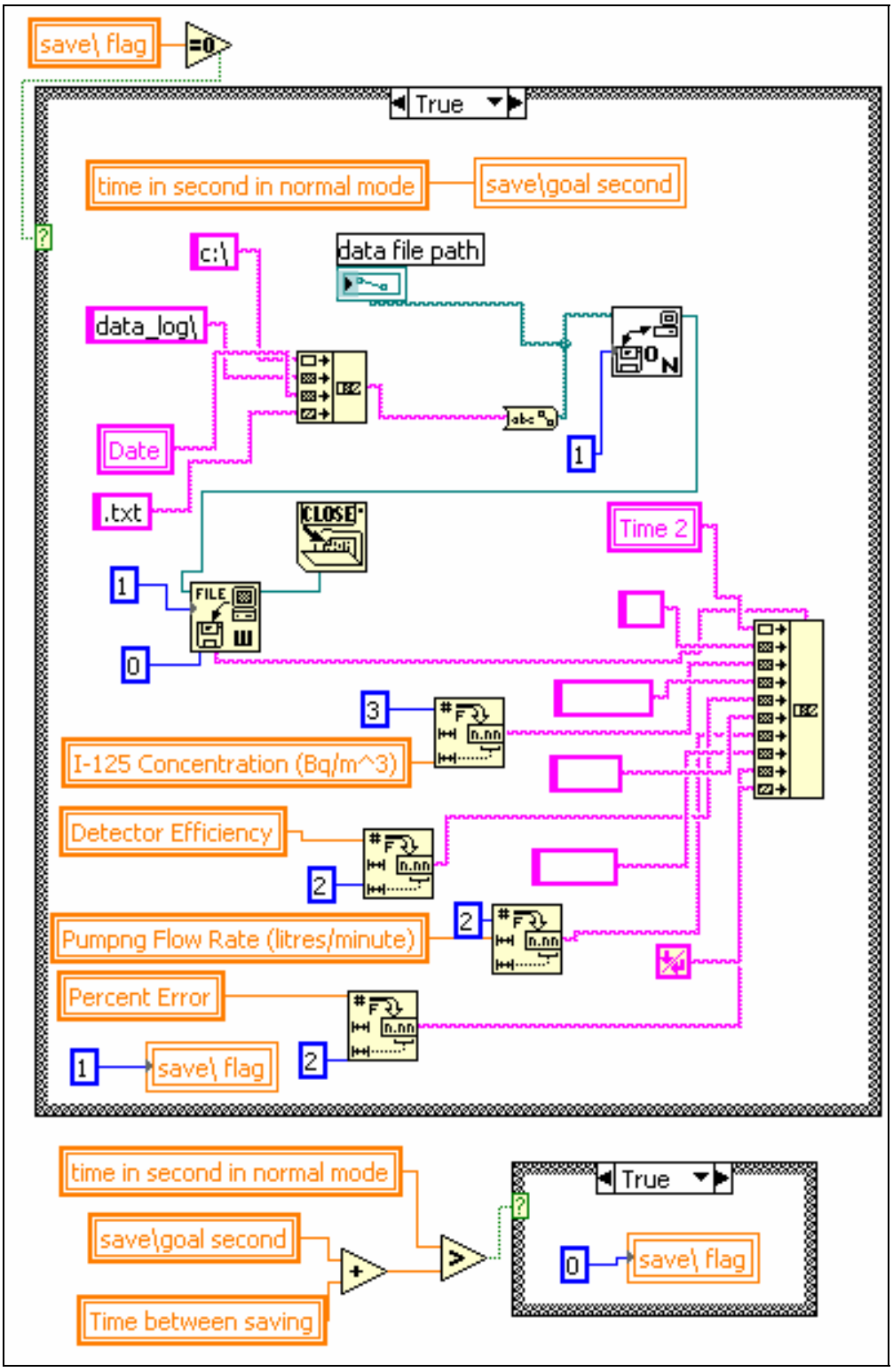


Figure 5-38: Writing data into a daily data file.

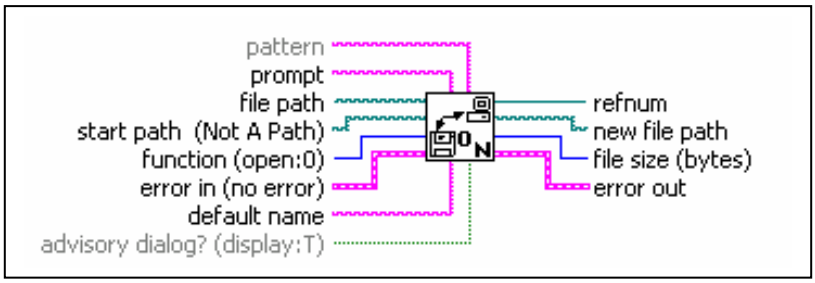


Figure 5-39: Open\Create\Replace file

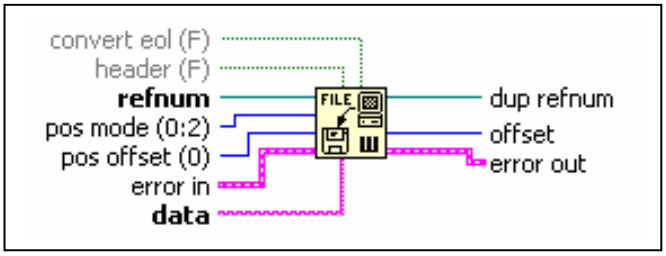


Figure 5-40: Write file

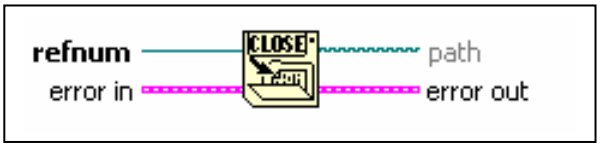


Figure 5-41: Close file

The first function, "Open\Create\ Replace file", must be adjusted depending on the status of the file. By default, this function is set for a file that already exists. If creating a new file is desired, "3" must be wired to the "function" as one of the input of the function of "Open\Create\ Replace file". However, wiring to "2" not only gives this ability to the "Open\Create\ Replace file" to build a new file, but also enables this function to replace an existing file. "4" is used only when the program needs to open an existing file to read the stored data. If "1" is connected to "function", the file can be either an existing or a new one. Depending on the purpose of using "Open\Create\ Replace file" and also the situation of the file, the different number is connected to the input of "function" in different process of the program.

The next function, “Write file”, shown in Figure 5-40, writes what is wired to the input called “data” into the file opened by the previous function. By setting “pos mode” together with “pos offset”, the place in the file to start the write (or read) operation can be specified.

The last function that is “Close file”, seen in Figure 5-41, is for closing the file that is specified by “refnum”.

The different kinds of data desired to be saved in the main file are shown in Figure 5-38. Firstly, the current time is saved. Iodine Concentration is then written in the file with a precision of 0.001. Although “Detector Efficiency” and “Pumping Flow Rate” are not the output data, they are useful for the future data analysis. Hence, they are written in the daily data files as well. As an input for the program, “Detector Efficiency” and “Pumping Flow Rate” can be varied by the operator adjustments. Their current values are recorded with the precision of 0.01. Percentage of error is another important parameter that is estimated for each calculated value of the Iodine Concentration. This output data is also written in the daily data files with the precision of 0.01.

I-125 Concentration, Detector Efficiency, Pumping Flow Rate and Percentage of error are all numbers. Therefore, in order to connect them to the “data” input of the “Write file” function, they all need to be converted to the fractional string. As seen in Figure 5-38, before concatenating input strings into a single output string, one function of ” Number To Fractional String” is used for each of these numbers. The precision of the output floating-point string can be adjusted in this function.

Figure 5-42 shows an example of the daily file. As shown in this example, the data is written from the third line. The first two lines are for the titles. Figure 5-43 shows the process of writing the first two lines.

First line specifies the location where the program is monitoring the iodine and also the date of the day. For this file in Figure 5-42 , EN means “Enclosure”.

Second line contains the titles for the proceeding data: “Time”, “Conc.(Bq/m^3)” that is referred to the Iodine Concentration, “Eff.” that is the abbreviation of the detector efficiency, “Flow Rate(L/min)” that means the pumping flow rate and percentage of error

(error(%)). At each time step, the complete data set is saved in the file. This time step can be adjusted by the operator but it is usually one minute. The numeric control of "Time between saving (seconds)" illustrates this time step. It is in the "Set up" page of the front panel.

As previously explained, during the initializing period there is no data for the Iodine Concentration. Thus, as it is seen in this file, instead of any data, just the starting and ending time of initializing period is recorded there.

According to the code of Figure 5-43, the first two lines, which indicate the location, date, and the titles of the data, will be written in a new file in the midnight of each day. This is the time when the date is changed and a new daily data file is created. In the case that the code is initiated in the middle of the day, these lines are added to the file at startup. As the process of this part is very similar to that in Figure 5-43, it is not repeated in a separate figure.

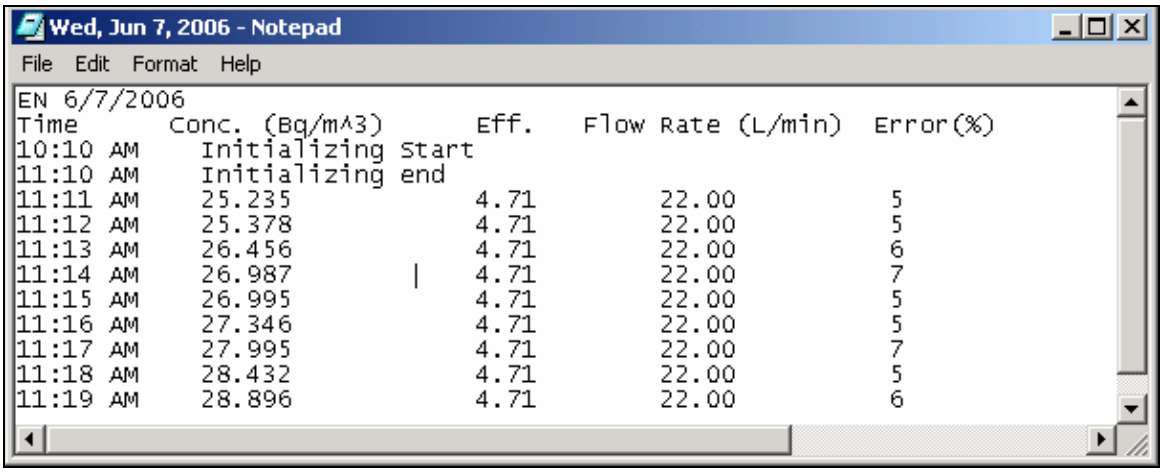


Figure 5-42: A demonstration for a typical daily file.

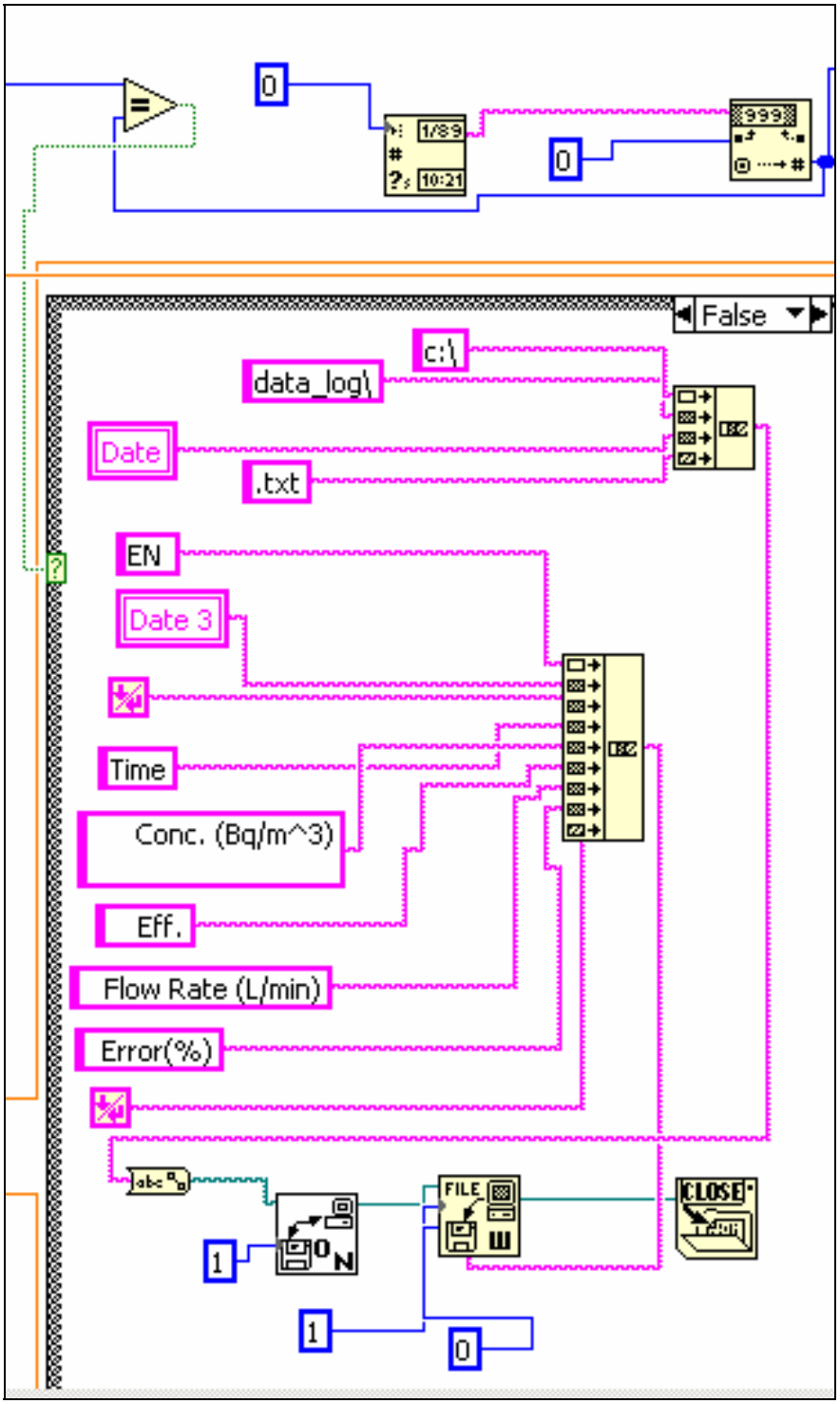


Figure 5-43: This part of the code adds the first two lines to the daily file.

5.6.1.2 "Eventlog" files

"Eventlog" files are the other series of files that are created in some special events. Unlike the daily data files, they are not created everyday. These files can contain any of the information listed below:

-The report of warning and alarm situation.

-Starting and ending time of filter changing.

-Report of entering to and exiting from the Demo status

-Record of changes in the main parameters that have direct effect in the amount of Iodine Concentration such as Detector Efficiency and Pumping Flow Rate.

-The errors occurring in the system.

If none of them happens in a day, the program will not have an "Eventlog" file of that day. It does not mean that for each of these situations there is a separate file. It will be created by a first condition that needs to be recorded in such files and then any other reports in that day will be saved at the same "Eventlog" file. The part of the code responsible for writing some of these conditions in this text file is considered below.

Figure 5-44 and Figure 5-45 show that any change in the value of "Detector Efficiency" and "Pumping Flow Rate" will be reported in the "Eventlog" files. These two parameters are important in calculating the Iodine Concentration and are often considered in the data analysis. As it is seen in these figures, the name of the "Eventlog" files included the date that it is created. Therefore, these files are shown as "*Eventlog.txt", where "*" indicates the date. Similar to the daily data files, "Eventlog" files are located in the "data-log" folder of "Local Disk(C:)".

While in "Demo" state the count-rates are not real, the amount of Iodine Concentration is completely different from real ones. Therefore, the amount of error will be very high at this state. For the future data analysis, it is necessary to keep track of the times that the code has been in the "Demo" state; otherwise, the fake data values are mixed with the real values of Iodine Concentration. Figure 5-46 and Figure 5-47 show the processes that record the entering and exiting times of the "Demo" state. This information is written in both the daily data file and the "Eventlog" file.

In order to call a file and write something into it, one may use the local variable of a file. However, this can result in some confusion and disruption in the code. The problem is that if the program reaches one of these local variables before reading their terminals, a dialog box opens asking about the location of the file from the operator, which is an interrupt in the program. Especially if the situation continues, the request for the location of the file will be repeated at each iteration. Therefore, the "local variable" can be used only if its terminal has been used in the previous steps of the program. Since each event can be the first one of the day, hence creating the "Eventlog" file of the day, in this part of the program using local variables must be carefully avoided, except in some special cases that are not discussed here.

Figure 5-48 shows a demonstration for a typical "Eventlog" file.

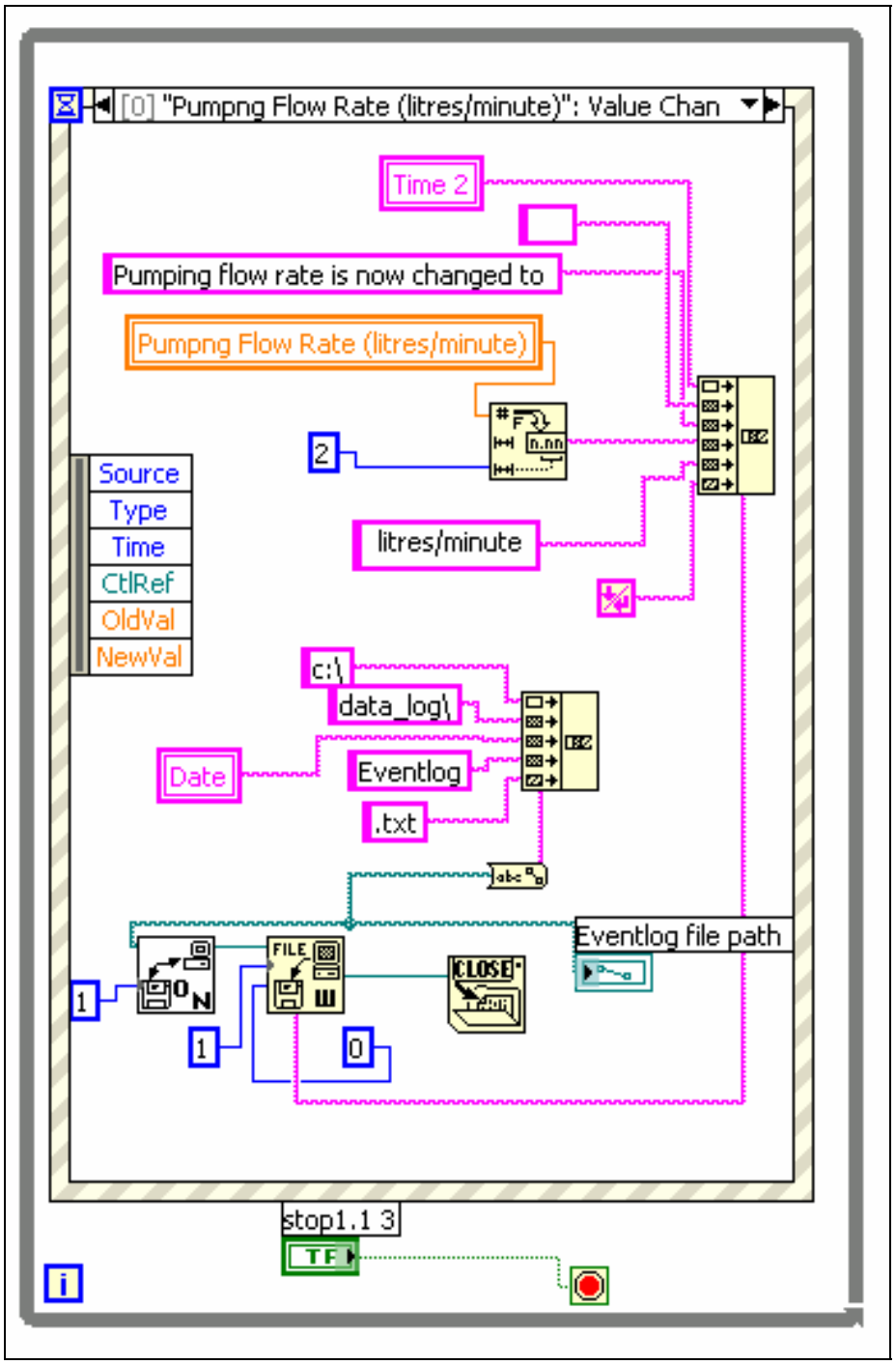


Figure 5-44: Changing the value of Pumping Flow Rate is reported in the "EventLog" file.

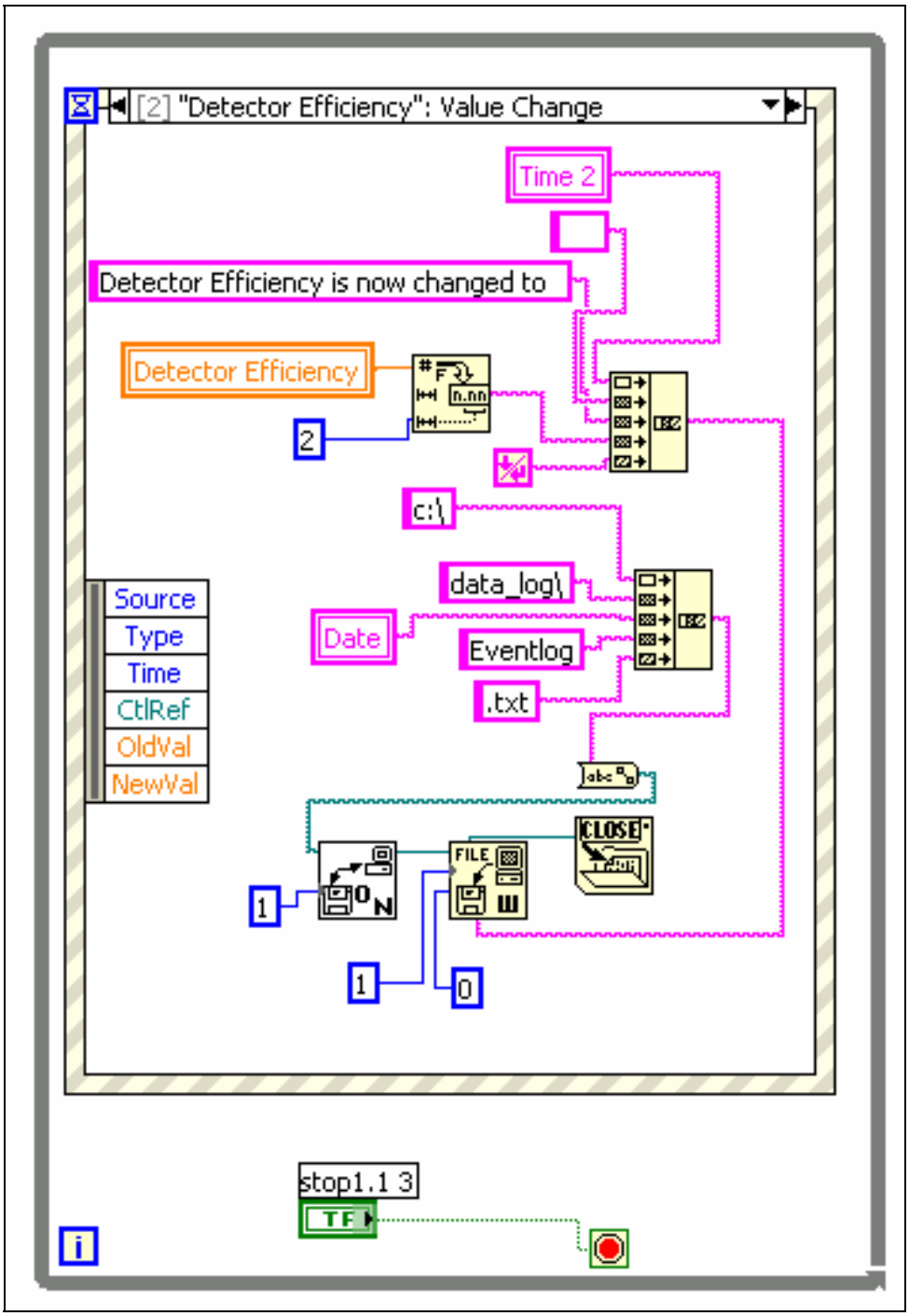


Figure 5-45: The updated Detector Efficiency is saved in the "Eventlog" file.

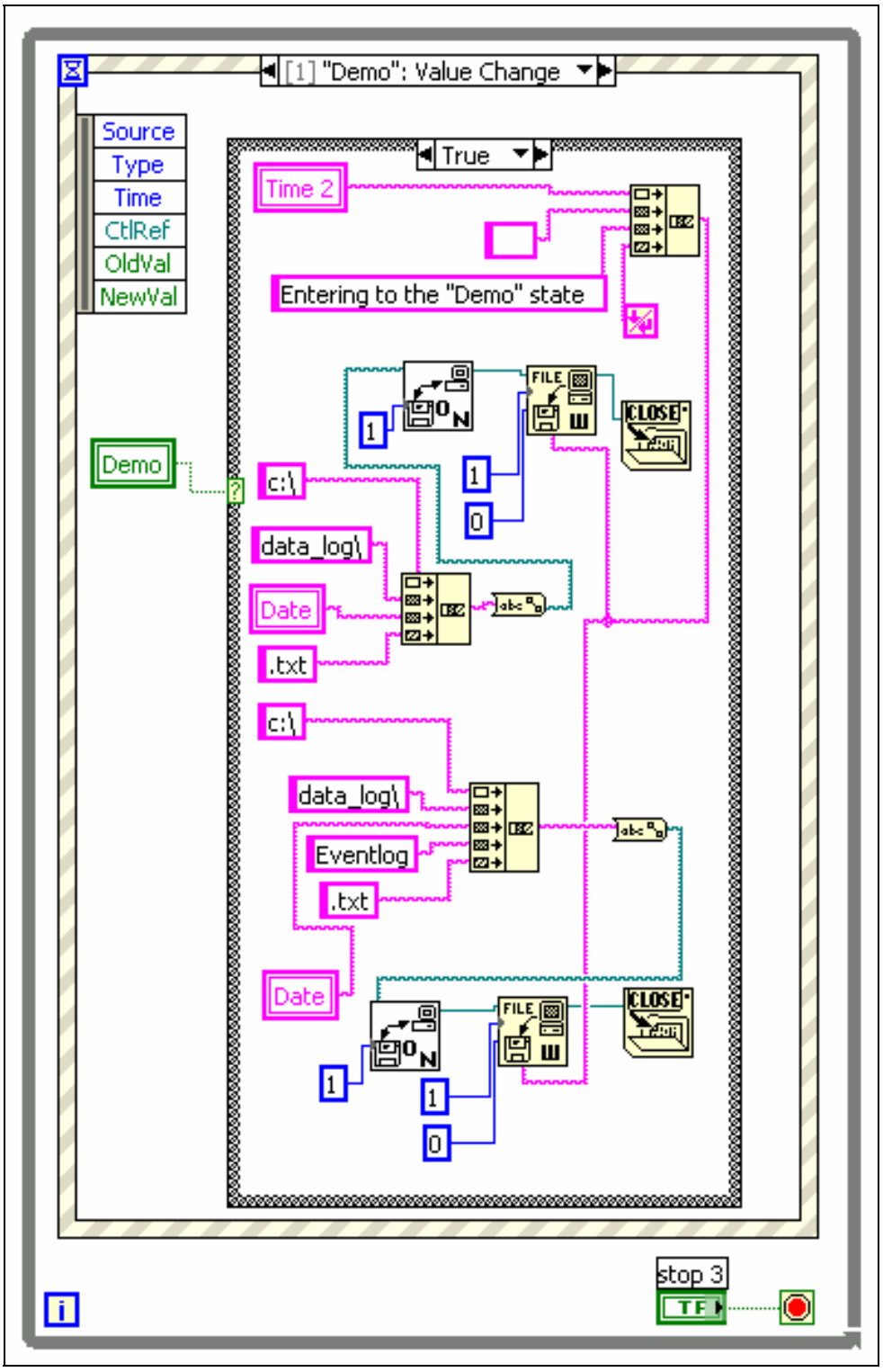


Figure 5-46: Entering to the "Demo" state is reported in the "Eventlog" file.

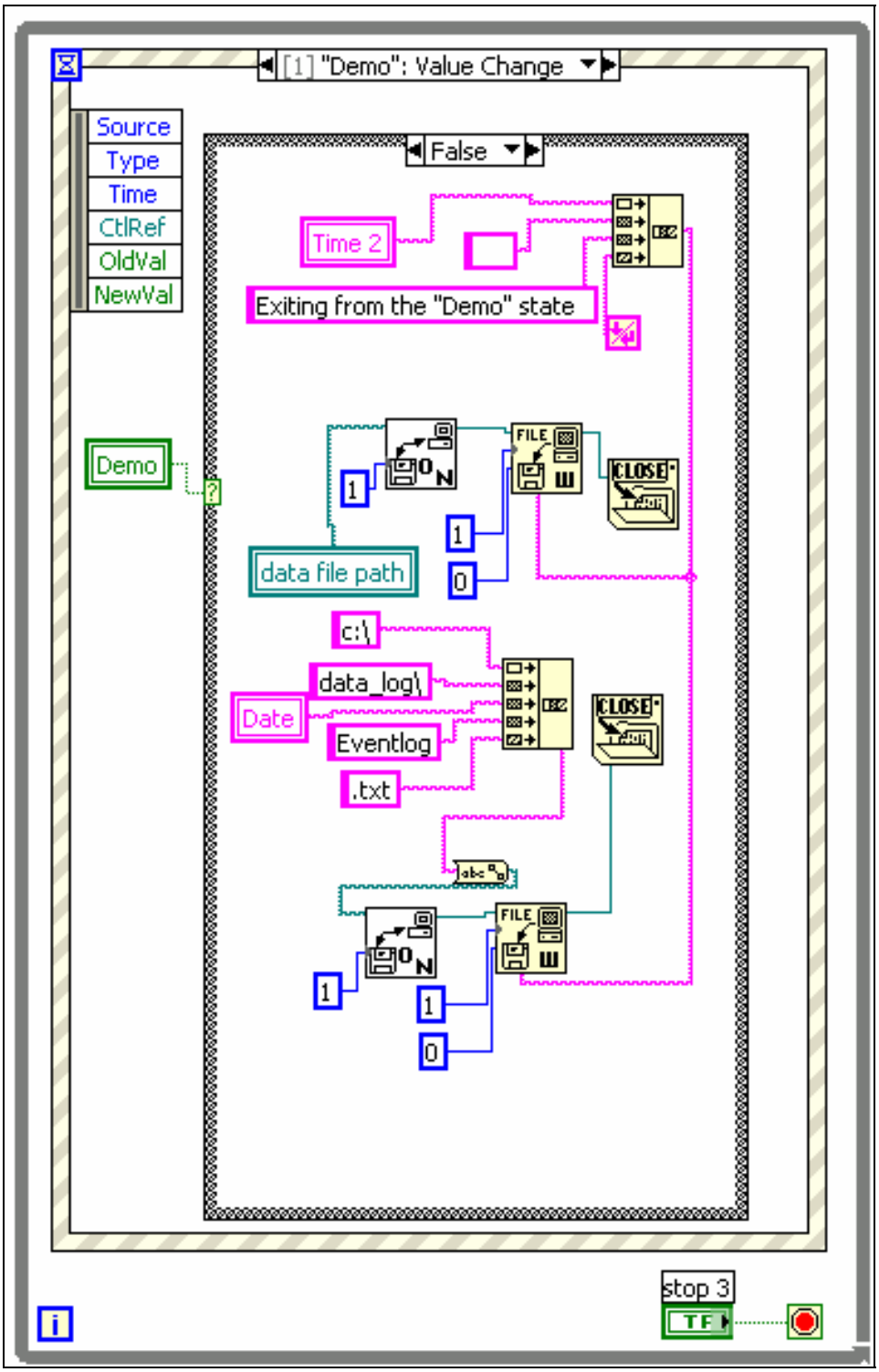


Figure 5-47: "Eventlog" file contains the exiting time of the "Demo" state.

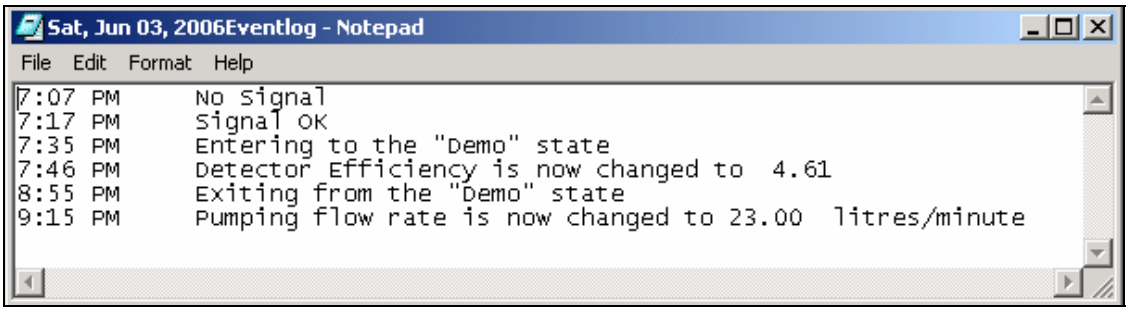


Figure 5-48: A demonstration for an "Eventlog" file.

5.6.1.3 "Logbook" files

"Logbook" files are used specifically to save the contents of the "Log Entry". This box that is shown is Figure 5-49 is basically a string control that the staff can manually write a note in. This manual entry is usually used to explain any abnormal situation for the system or any occurrence in the reactor that can affect the amount of iodine-125 in the air.

In order to save the note to the "Logbook" file, the button at the bottom of the box must be clicked. Pushing this button also clears this entry. Any other messages in this box at the same day will be recorded at the same file.

Figure 5-50 shows the process of creating a "Logbook" file, saving the content of the "Log Entry" box into the file, and finally clearing the box. It is seen that the creation date of the logbook file is also contained in its title.

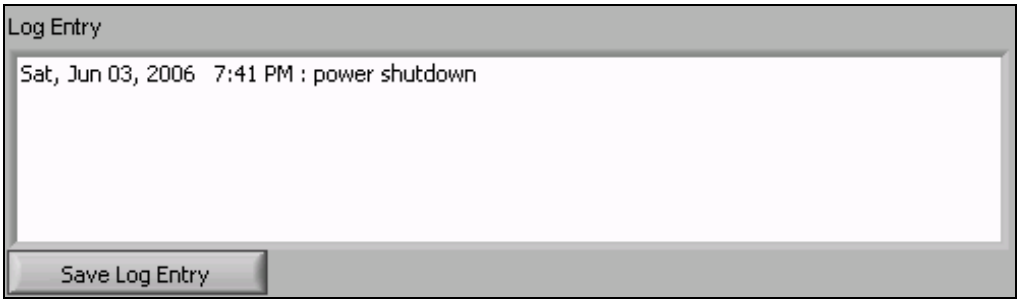


Figure 5-49: A demonstration for the "Log Entry" box in the main screen.

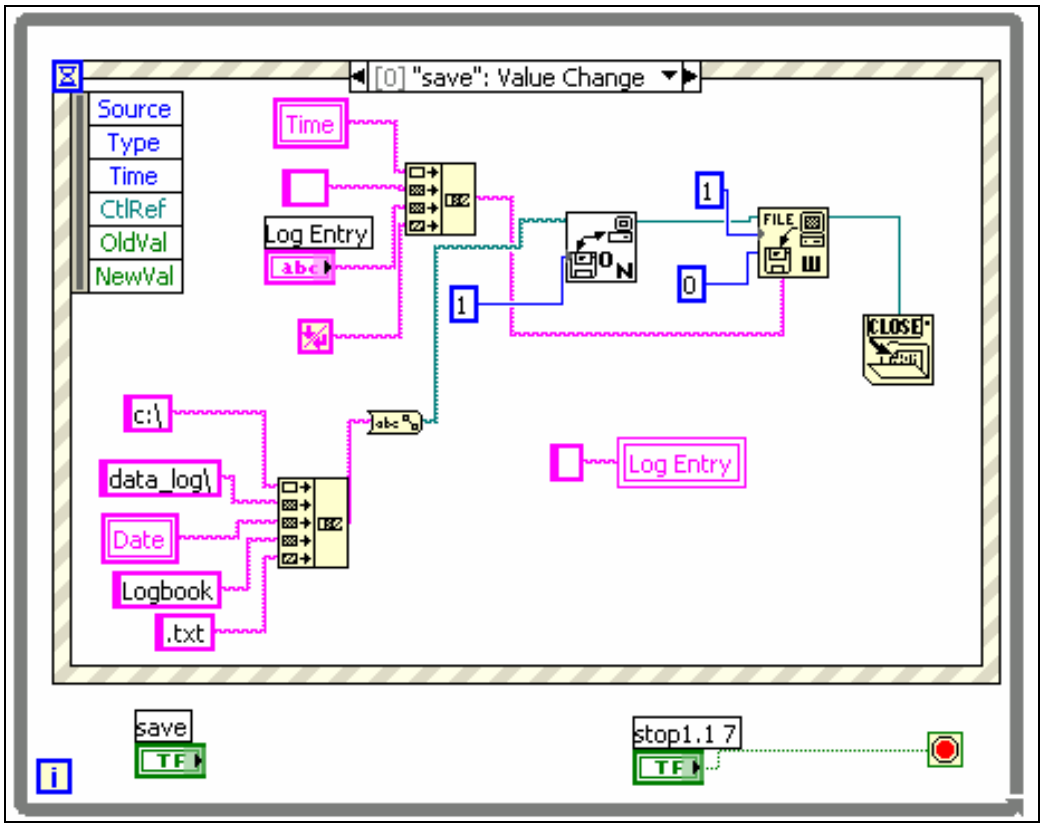


Figure 5-50: This code shows these processes: 1) Creation of a "Logbook" file or opening an existed file. 2) Then saving the content of "Log Entry" box into this file. 3) At last clearing this box.

5.6.1.4 Other files

The three groups of files that were described above are the main files that provide data for data analyzing. The program also contains some other files that are usually created to save some variables used during the execution of the program. For example, if the program stops running for any reason, the last changes in the indicators must not be lost. By saving these changes into a file and then reloading them at the beginning of the next execution, the program has the ability to continue its job without a need to update the indicator data, or without losing the data. For instance, suppose the Pumping Flow Rate is changed in the middle of the execution of the code. LabVIEW cannot change the

default value when the program is running. Thus, if the new value of the pumping flow rate does not reload from a file at the start of the next execution, it will return to the default value of the Pumping Flow Rate, which is not the updated value, and hence the operator should manually fix it. Therefore, the program saves the new value of such parameters in a file, reads this value at the start of the execution, and updates the indicator. These parameters are listed as below. The program has a separate file for each of them:

-Time between saving

-Queue

-System Initializing Period

-Detector Efficiency

-Pumping Flow Rate (litre/min)

-Sample Length

-Mail Server

-Recipients

-Email Subject

-Email Message

-Red Light and Audio Threshold

-Amber Light Threshold

-Silence Audible Alarm

-Y Axis Minimum

-Y Axis Maximum

-Password (contains four parts: Pass, Enter present password, Enter New Password, Reenter New Password)

The name of each file is according to the name of each parameter. These files are all located in the folder of "Variables" in the Local Disk (C:). As an example, writing into one of these files is shown in Figure 5-51. This figure shows that when the value of "Time between saving (second)" is changed, it is written in a file with the path of "c:\variables\Time between saving (second).txt". The value that is connected to the

"function" input of "open\create\replace file" is "2" which means that an existed file is replaced by a new file. Since in this procedure only the last value is needed, the old value is replaced by the new one. When the input of "function" is wired to "2", the VI always displays a dialog box and asks if the new file should replace the old file. To avoid this interruption in the program, this step is inactive by connecting a "False" constant to the input of the "advisory dialog?"

In the next part of this section, reading from this file at the beginning of the running will be explained.

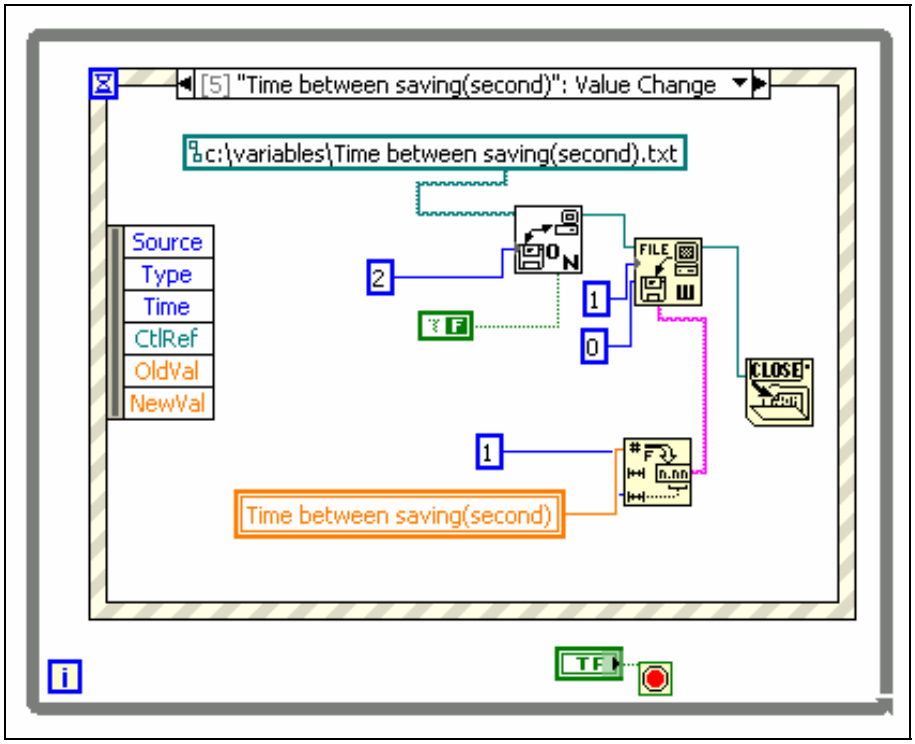


Figure 5-51: The old value of "Time between saving (second)" is replaced by the new one if the operator changes its value when the program is running.

There is also another file that is called "writeweekly". This file is created for saving the array of data that is used for rebuilding the graph of "Long Term Average I-125 Concentration (Bq/m^3)".

5.6.2 Reading from Files

In order to access the data in a file, one needs to read from the file using a proper function. There are several different ways in LabVIEW to read from a file. In the method applied in MASIS code, the functions used for reading is very similar to the functions employed for writing. All the series of writing function are used for reading except "Write file". Instead of this function, "Read file" is employed. The function of "Read file" can read from a file that is opened by the function of "Open\Create\Replace file". The file is identified by the input of "refnum". The "Read file" is shown in Figure 5-52. "Pos mode" together with "Pos offset" locates the place where the reading of the file is started. The format of the specified file has effect on how the file will be read.

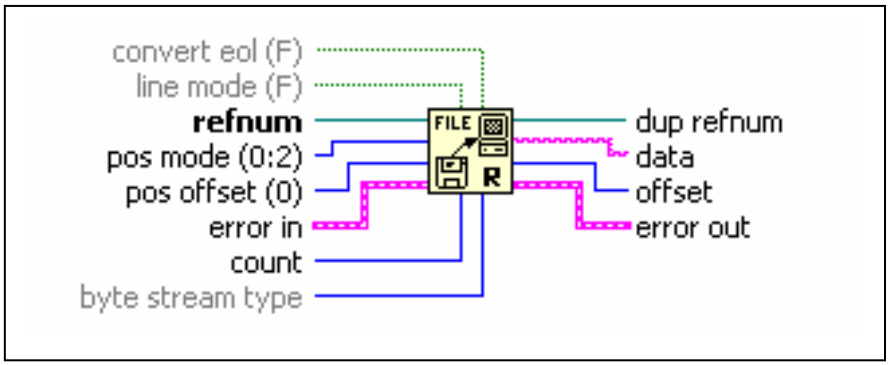


Figure 5-52: The "Read file" used in the reading process from a file.

Also, there is an extra function used in the reading process. This function does not exist in the series of functions used in the writing process. The function that is called EOF (end of file), sets the logical end of file that is specified by "refnum". Figure 5-53 shows the appearance of this function.

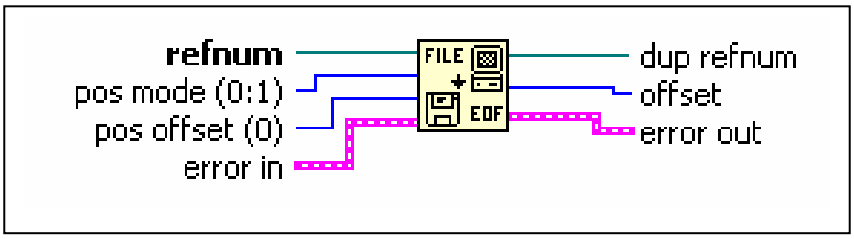


Figure 5-53: Function of EOF (end of file)

As an example of reading from a file, Figure 5-54 shows the process of reading the value of "Time between saving (second)" from the corresponding file.

One who reviews the MASIS code may encounter questions about some processes at the first iteration of the code. These include *apparently* unnecessary reading and writing processes for different variables. For simplicity, only such a process for one variable, i.e. "Time between saving (second)", is presented and its necessity is discussed here.

Until now, it has been seen that if the value of "Time between saving (second)" changes during the run time of the program, the new value is saved into a file. If the program stops running and exits from the LabVIEW environment for some reason, the last value of "Time between saving (second)" is loaded from the file at iteration #1 of the next run. What happens if the operator changes the variable before running the program? In this case, the variable has a new value only at iteration #0 (i.e. the first iteration). But immediately at iteration #1, as seen in Figure 5-54, this value is replaced by the data already written in the file i.e. the old value of "Time between saving (second)" .Note that when the operator adjusts the variable when the program is not running, the VI does not realize the change. Therefore, the process that is shown in Figure 5-51 is not applied for such a situation and the variable remains unchanged in the file.

To avoid this problem and also to save the new value of the variable into the file, the process shown in Figure 5-55 is added to the code. This process is run at iteration #0, i.e. the first iteration. The program reads the information from the file at iteration #1. The default value for "Time between saving (second)" is 60 seconds. The variable gets its default value upon the program startup unless it was manually changed prior to the run. Therefore, only when at the first iteration (iteration #0) the value of "Time between saving (second)" is not equal to default value (60 seconds), the process of Figure 5-55 must be performed. As it is seen in this figure, there is a flat sequence structure. In the first sequence the value of "Time between saving (second)" is read and put into another indicator that is called "for comparison of time between saving". In the next sequence, if this indicator is different from the "Time between saving (second)", the new value

replaces the old value in the" Time between saving (second)" file. So before reading from this file in the next iteration, the file is updated. Also, for the next execution of the program the new value will be reloaded from the file.

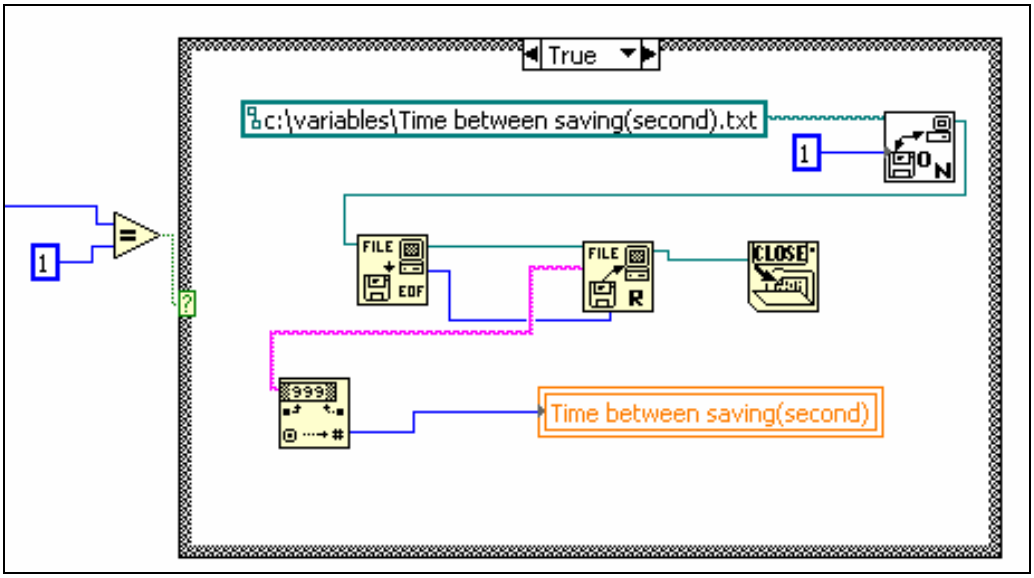


Figure 5-54: At the iteration #1 (second iteration), the variable of "Time between saving (second)" read the value saved in the corresponding file.

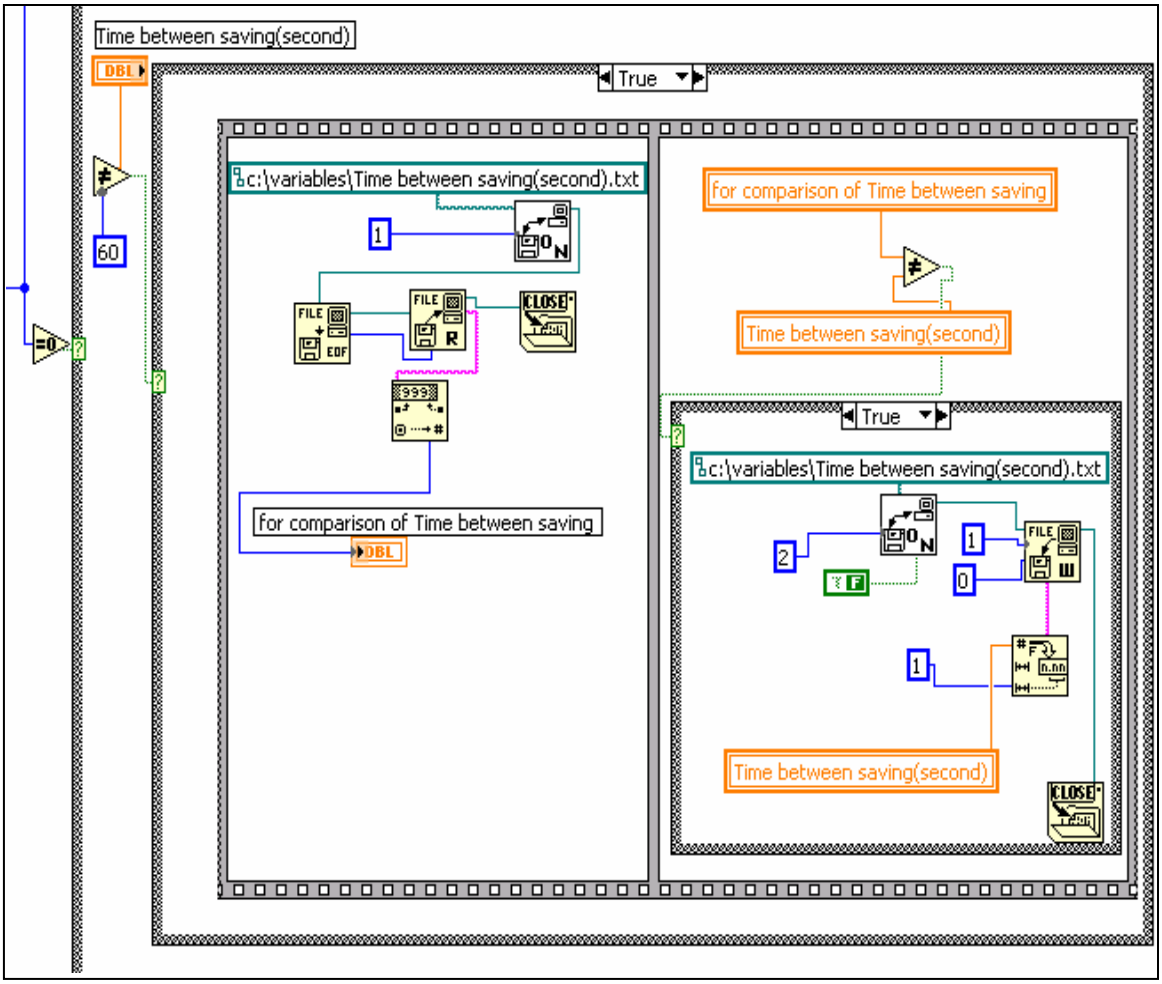


Figure 5-55: This process runs in iteration #0, i.e. the first iteration. It is designed for the situation that the operator changes the value of a variable when the program is not running, and therefore, the new value is not saved in the file.

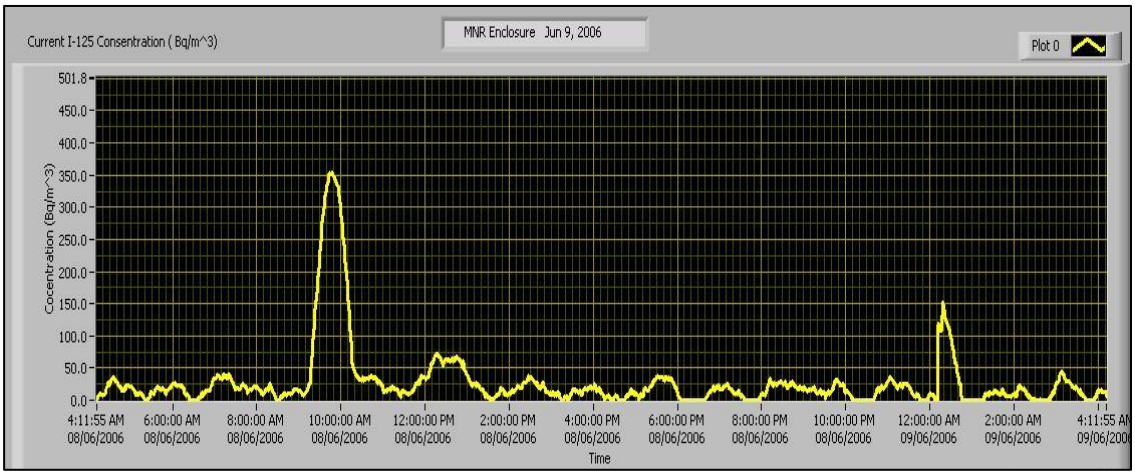
5.7 Plotting the Graphs

The next step is transferring the data to the graphs. There are two graphs in the main screen. One plots the current Iodine Concentration in the last minute: "Current I-125 concentration (Bq/m^3)". The other graph shows the average Iodine Concentration of each filter: "Long Term Average I-125 Concentration (Bq/m^3)".

There are different kinds of graphs and charts available in LabVIEW. But for our purpose the most suitable one is the "XY Graph".

The appearance of the Current Iodine Concentration graph and part of the code for transferring data into it are seen in Figure 5-56 and Figure 5-57 respectively. Figure 5-58 shows the appearance of the graph that indicates the average of Iodine Concentration of each complete period of Normal Mode that is between two filter changes. As filter is usually changed every week, this graph is called "Long Term Average I_125 Concentration (Bq/m^3)". Figure 5-59 is part of the code corresponding to the long term average graph.

It is necessary to use a buffer for transferring the data into a graph. Without buffer, only the current point is available and previous ones will be lost. A larger buffer provides access to more points of the previous data. For this code, "20000" points is a good choice for the size of the buffer of each graph. This size does not make the program slow while it has access to the reasonable points in the graphs. As it is seen in both Figure 5-57 and Figure 5-59, the "XY chart buffer" is employed to apply this buffer. The appearance of this function is shown in Figure 5-60.



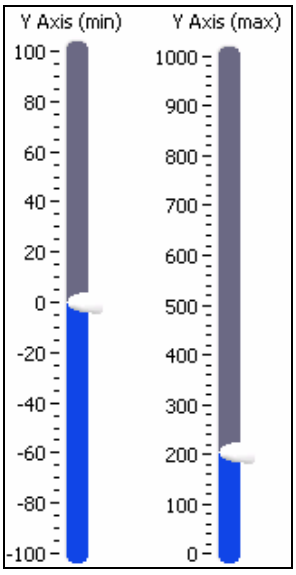


Figure 5-56: The appearance of the "Current I-125 concentration (Bq/m^3)"graph and the vertical pointer slides that the operator can adjust the minimum and maximum of the Y-axis of the graph by them.

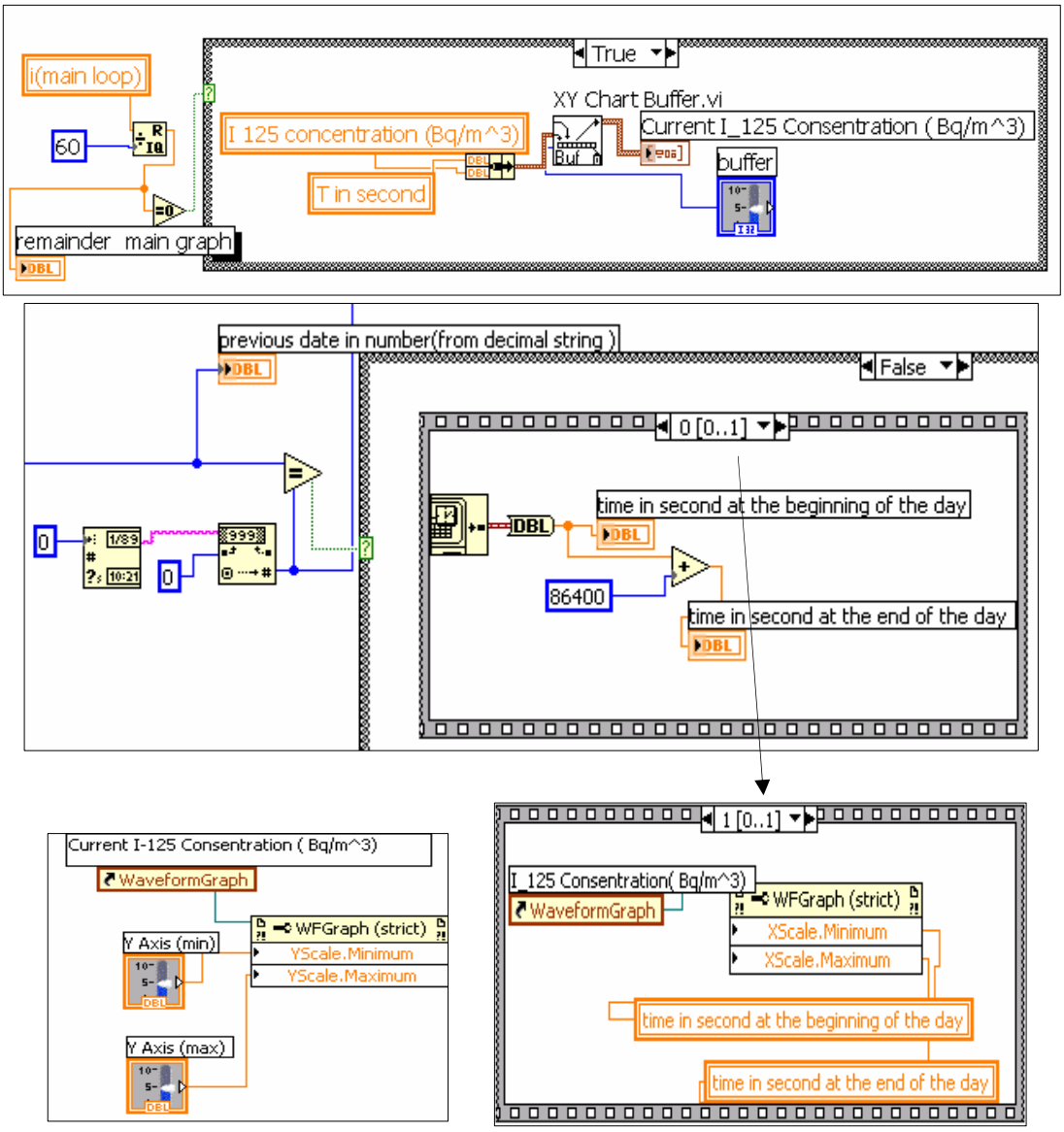


Figure 5-57: Parts of the code that control the "Current I-125 concentration (Bq/m^3)"graph and transfer the data into it.

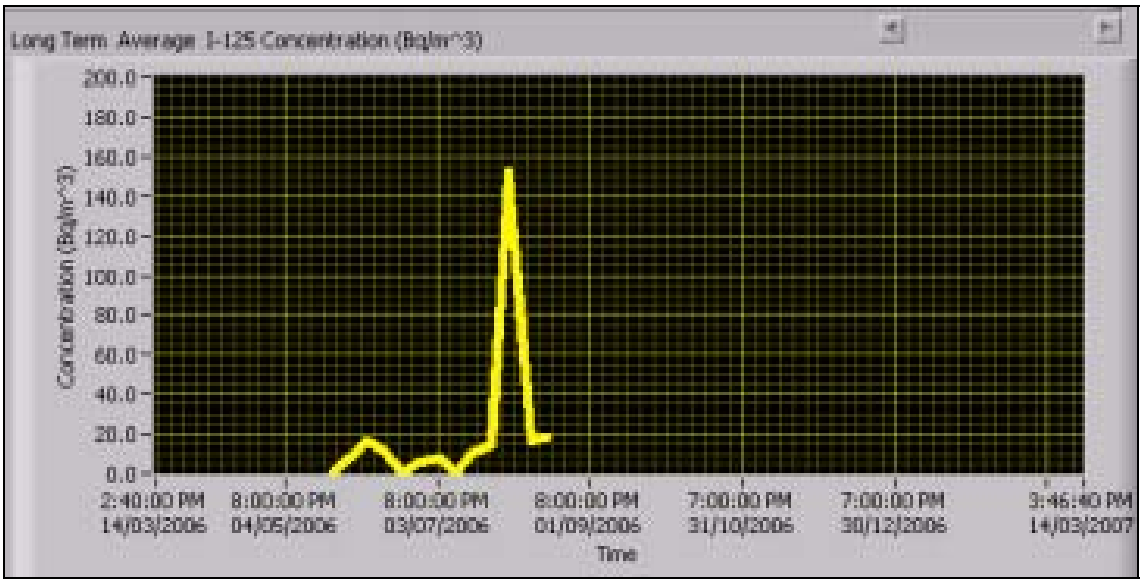


Figure 5-58: The appearance of "Long Term Average I-125 Concentration(Bq^3)"

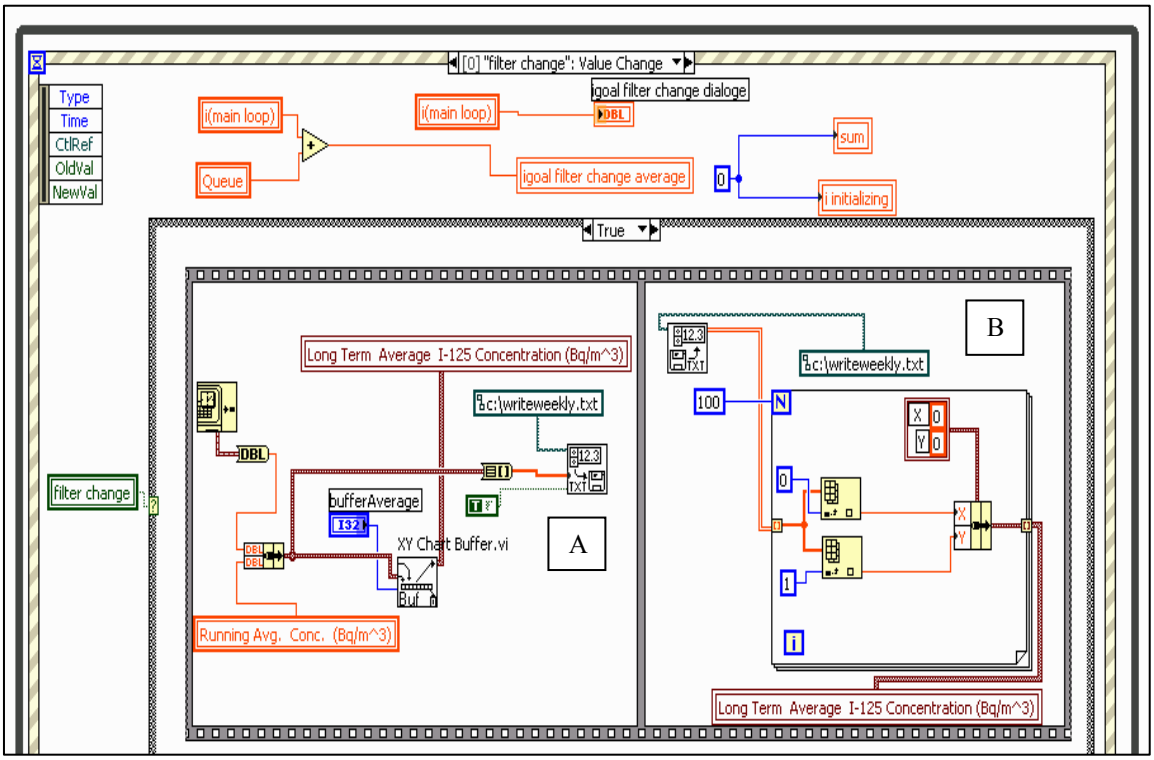


Figure 5-59: Part of the code related to the data transferring into the graph of "Long Term Average I-125 Concentration (Bq/m^3)"

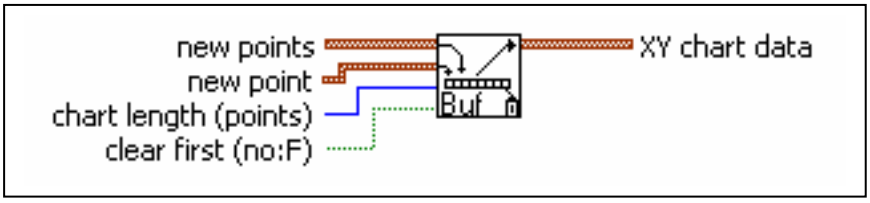


Figure 5-60: "XY chart buffer".

Although "XY chart buffer" is a LabVIEW sub-VI, it is likely that it is not found in some packages of software. Therefore, its code is included in Appendix 8. It is also possible to make such a buffer by mimicking the process shown in its code.

According to the code of both graphs, the primary input of the X axis is the current time in second. The format and precision of the current time is converted to the "absolute time" to show the time and date in the X axis similar to the graphs of Figure 5-56 and Figure 5-58. In the graph of "Current I-125 Concentration (Bq/m^3)", twenty four hours is seen in the X axis, which is changed at midnight. However, the "Long Term Average I-125 Concentration (Bq/m^3)" graph has one year X axis limit. These axes change on a daily and yearly basis respectively. The process is as follows:

At the first run of the code, the "Current I-125 Concentration (Bq/m^3)" graph shows the time from the present time to the next twenty four hours on the X axis. However, at the midnight, it changes from one midnight to the following one. In the "Long Term Average I-125 Concentration (Bq/m^3)" graph, at the start of the program, the X axis shows the absolute time of the present time up to a complete one year afterwards. At the New Year, it changes to the whole current year. It is always possible to scroll the graphs through the x axis scroll bars and see the data of the previous days or years.

The Y axis shows the amount of Iodine Concentration in both graphs. The Y axis of "Long Term Average I-125 Concentration (Bq/m^3)"graph has an auto scale. However, since for the "Current I-125 Concentration (Bq/m^3)" graph, this was not desired by the operator, the auto scale was turned off and changed to adjustable from the display. The operator can change the scale of Y axis by two vertical pointer slides in the password

protected area of the front panel. They are shown in Figure 5-56. The Y axis of this graph has a default minimum of "0" and maximum of "200" Bq/m^3.

One of the important tasks of the "Long Term Average I-125 Concentration (Bq/m^3)"graph is reloading all previous data when the program runs again. It is important because in case of stopping the program, exiting from the LabVIEW environment or any sudden crash of the PC, no previous data point of average Iodine Concentration must be lost in the graph. In order to achieve this assurance, the program saves the array of data into a file and reloads it at the beginning of the new run. However, if someone traces the main code of MASIS, s/he may become confused about some of the steps. Reloading data from a file is not a straight forward job for the "XY graph". As it was mentioned at the beginning of this section, there are some different kinds of graphs or charts in LabVIEW used for demonstrating data. Among them the "XY graph" is chosen because of some of its capability to have a better display. But actually if we want the graph itself to remember the previous points then we need to use a "Chart" instead of a "Graph". Charts have an option for chart history length but graphs do not have this ability. The "XY Graph Buffer" gives us similar functionality by holding previous data points but the graph itself is being completely rewritten every time (as opposed to the chart which appends its data). It means all the previous data will be lost just after having a new point. The scenario is that: the array of data is saved into a file and then at the beginning of the execution of the program, the data is read and translated to the graph. But when the first new data is arrived into the graph, all the previous data that was from the file is deleted.

As it might be noticed in the Figure 5-59, this problem is solved by an additional step that is marked by "B". This step shows that the program reloads the whole data from the file after getting each new point (in a sequence, so the process is not visible during the execution).Since the new data in the program (one point in the graph) is coming once every week or so to the graph, It doesn't make the program slow and also even after several years the file will not be big. Moreover, at the time that the program reloads the graph from the file, the process is in a semi pause as the operator is changing the filter,

which usually takes about fifteen minutes. Therefore, the iteration time on that period is not concerned either. Hence, the trick works and no data point is lost.

Step "A" in Figure 5-59 Shows saving each data point in a file. As it is seen there, the name of this txt file is "writeweekly" in local disk (C:). All the data is saved in the file in a two dimensional array format. One dimension is the time and the other is the amount of average Iodine Concentration of the filter removed at that time.

Figure 5-61 is part of the code for reloading all data from this file to the graph at the start of running the program.

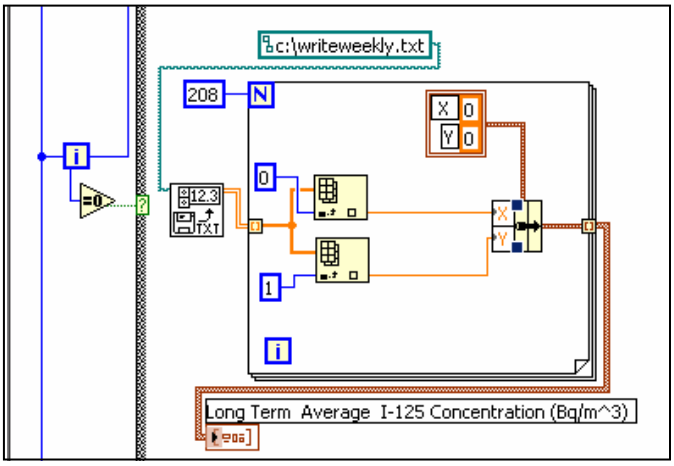


Figure 5-61: Reloading the array of data from the "c:\writeweekly.txt" to the "Long Term Average I_125 Concentration (Bq/m^3)"graph.

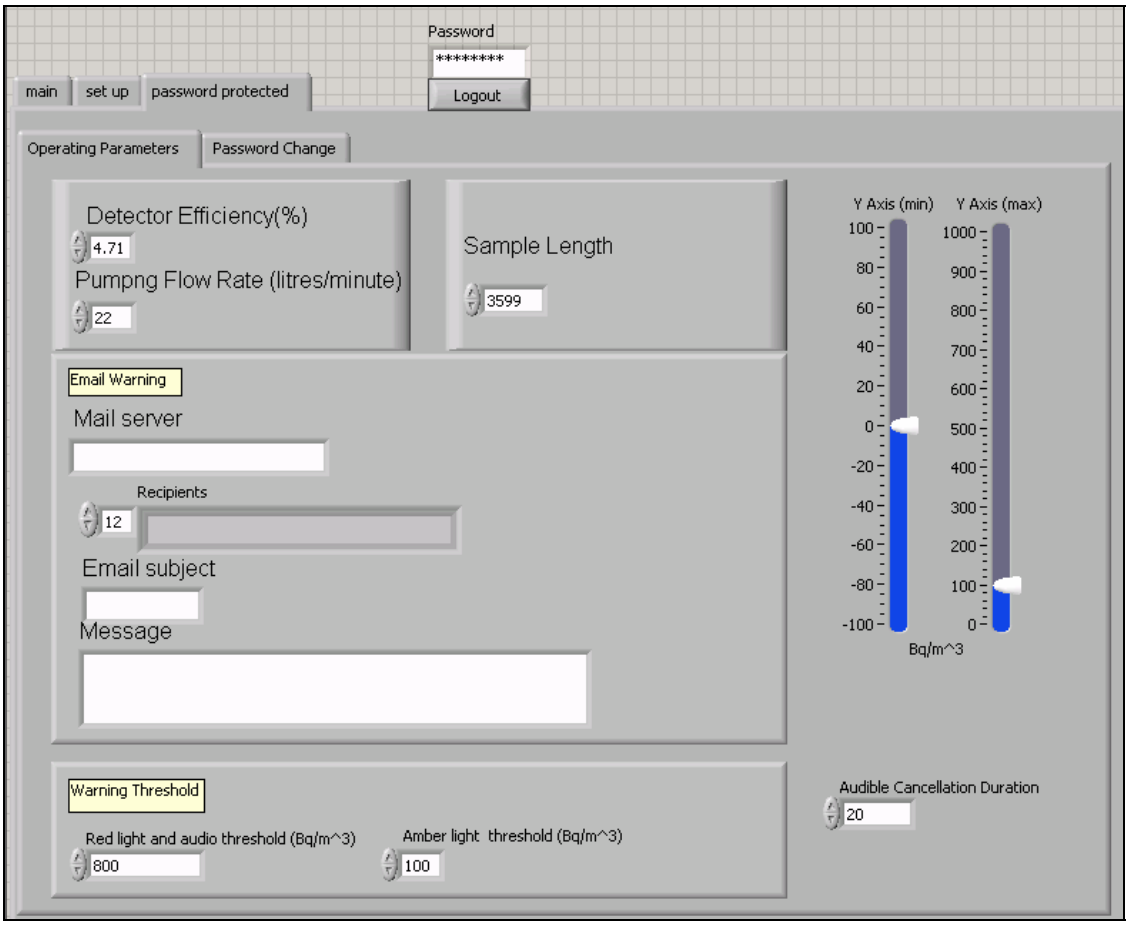
5.8 Making and Controlling the Password Protected Area

One page of tab control in the front panel is password protected. As shown in Figure 5-62 this page contains some adjustable parameters of the program. Changing the password is also possible in this area.

The process to make this area protected by a password is shown in Figure 5-63 . As seen in the code, for safety purpose, it is automatically logged out if there is no activity in that area within one minute.

The password can contain letters and numbers with any length. In order to change the password, the operator must first enter the present password as shown in Figure 5-62. If the entered password is wrong, a dialogue box opens and asks to try again. Then, the new password must be entered two times. In case of mismatch between these two, a dialogue box opens and asks to re-try.

The program has a default password for its first run on a new computer. If the operator changes the password, it is saved in a file and read from it at the beginning of the next run. The main part of this process is shown in Figure 5-65.



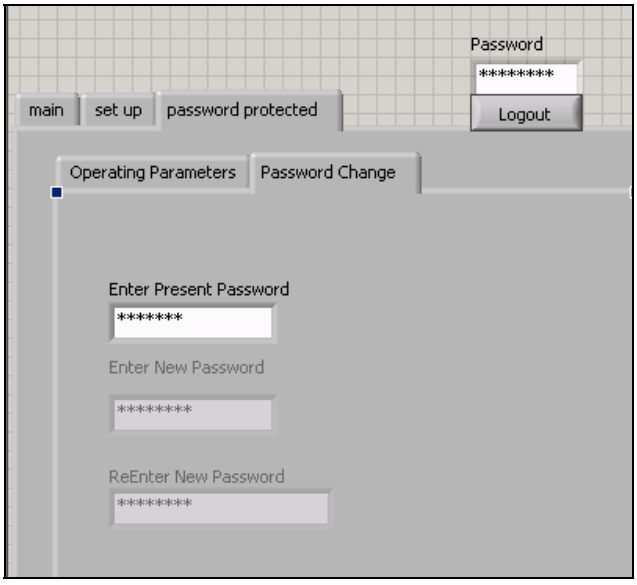


Figure 5-62: The front panel of the password protected area. This area has two pages; one for adjustable parameters and one for changing password.

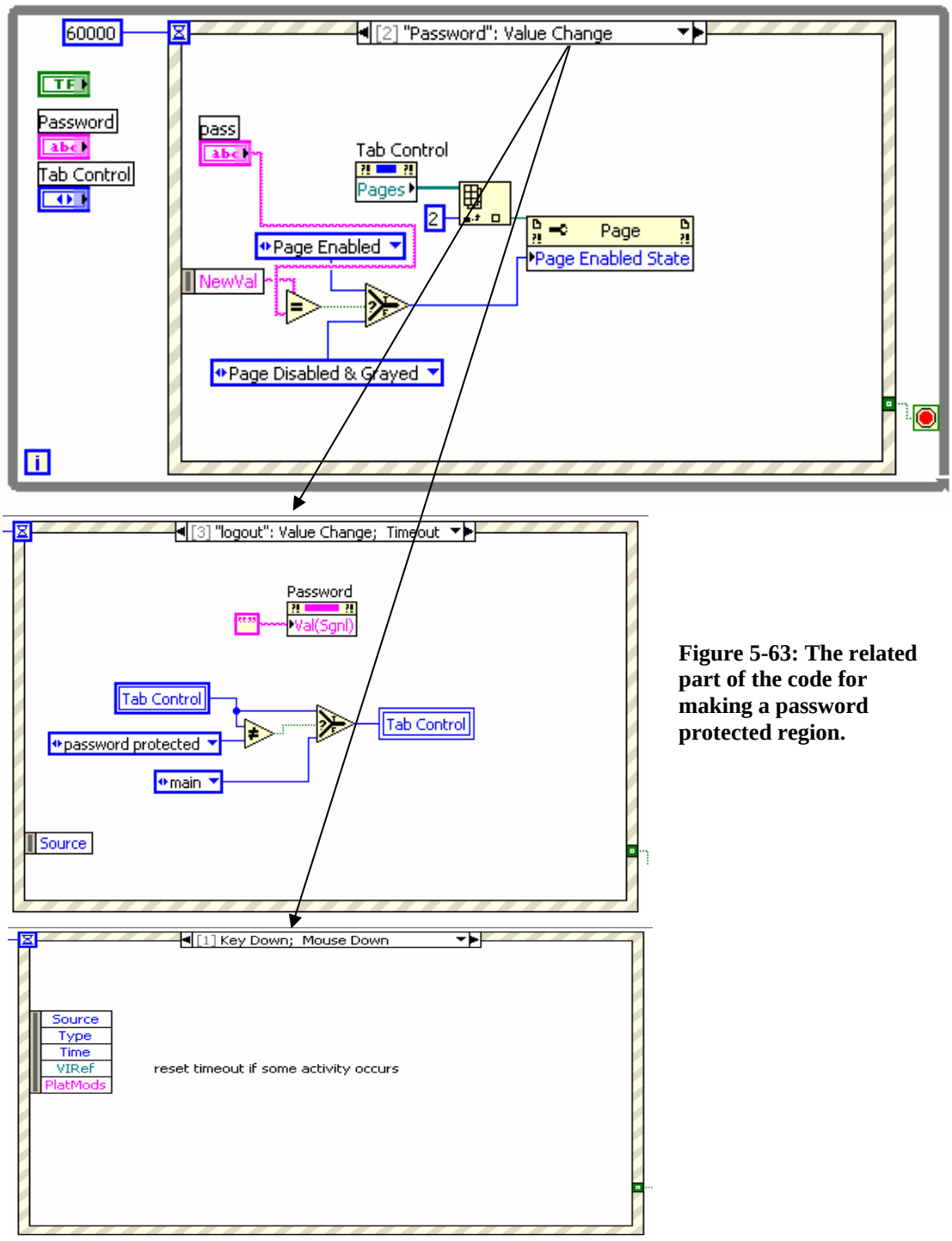


Figure 5-63: The related part of the code for making a password protected region.

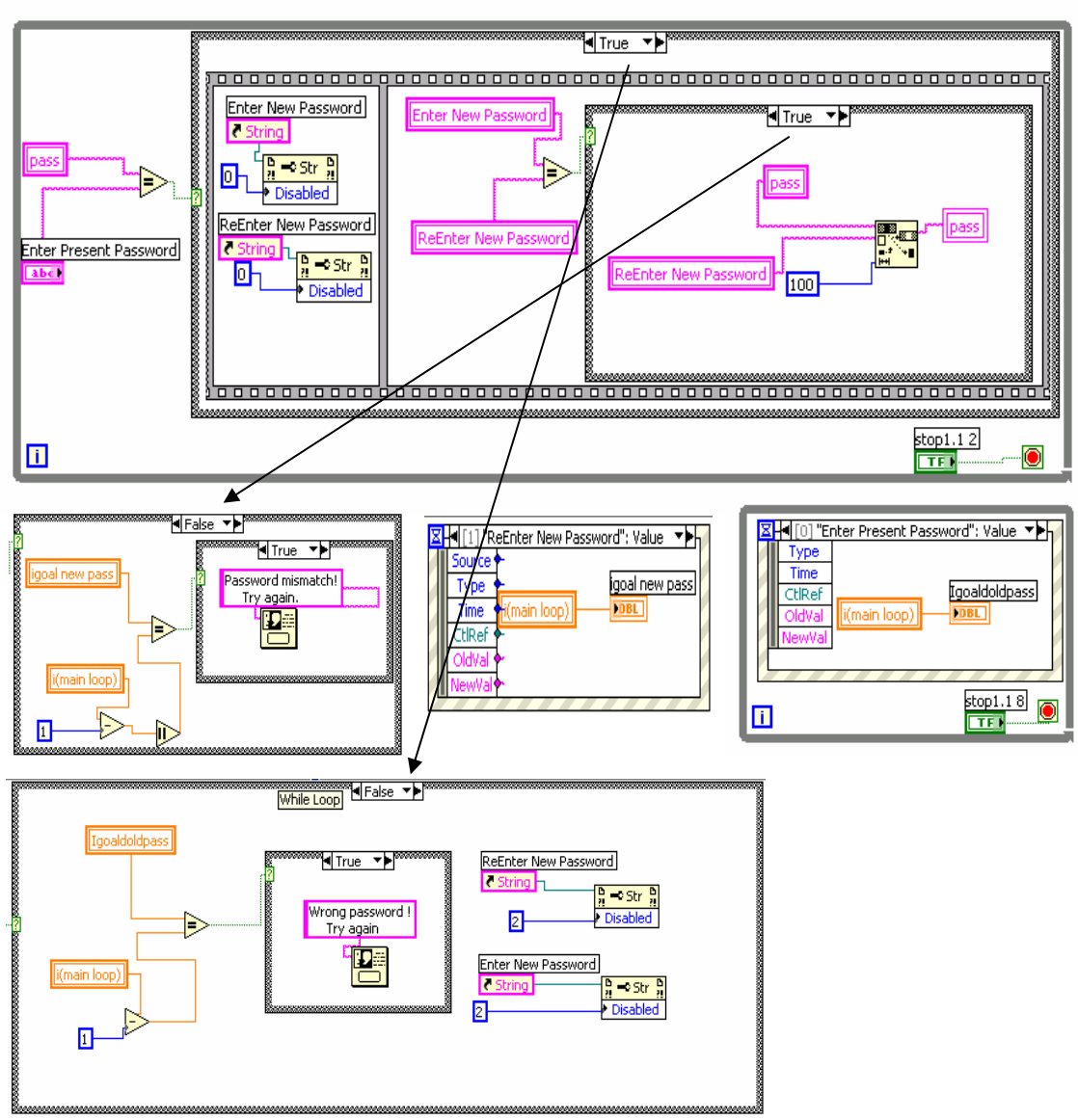


Figure 5-64: Part of the code related to changing the password

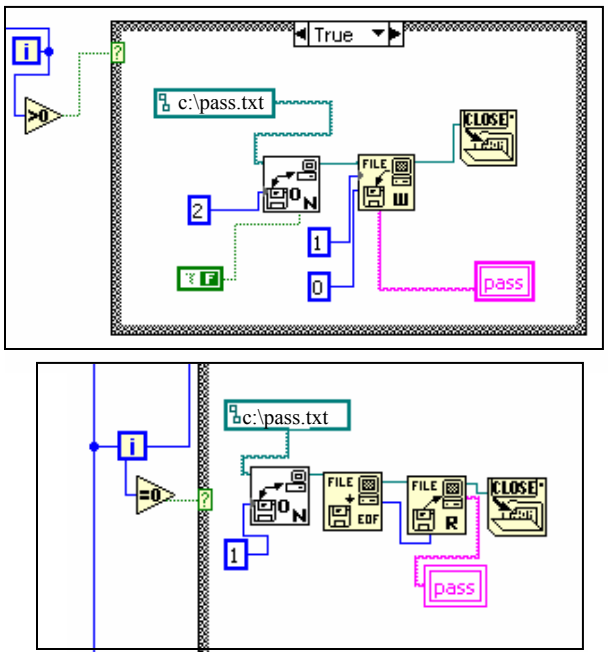


Figure 5-65: Write the password into a file and reload it at the beginning of the program.

Chapter 6

Conclusion and Recommendations for the Future Works

Iodine-125 is one of the radioactive isotopes produced at the McMaster Nuclear Reactor (MNR). It is used especially for thyroid treatment and prostate cancer therapy. However, chronic doses of small quantities of I-125 for workers can harm the body especially the thyroid organ. I-125 gamma photons are very difficult to detect since they have a very low energy. A suitable detection system is required to enable the monitoring of very small concentrations of I-125 in the air.

Due to some technical and practical problems with the old monitoring system, a new system called MASIS was designed to improve the reliability of monitoring and advising system for the concentration of I-125 released to the air. In addition, MASIS provides the users with some new features both in monitoring and in alarm and annunciation system.

This thesis presents the outline and the details of the software and hardware for the new system. Software is a computer program, with appropriate peripherals run within the LabVIEW 7.1 environment to analyze the data coming from the hardware. Hardware includes an air pump, an NaI detector, a charcoal filter, LabJack U12, PC and a Signal Tower. The hardware counts the iodine-125 gamma photons and passes the information to the LabVIEW program. It also sends the proper control signals to the alarm and annunciation system. All these are to give the nuclear reactor operators a computerized, simple, safe and flexible monitoring facility.

For future improvement of the system, it can be helpful to add a feature to the alarm and annunciation system to automatically make telephone calls to the reactor crew in the alarm situation. It was initially planned to add this feature in MASIS, but due to some funding barriers, the remote annunciation was limited to auto-sending email messages.

Also for the hardware, the present detector can detect all the photons between 10 KeV and 100 KeV. Therefore, the detector may detect not only iodine but also other elements in this energy range. This is one source of inaccuracy and could lead to an overestimation of the Iodine Concentration. There can be possible improvements in using a narrow band detector centered at 33 KeV, which is I-125 gamma photons energy, to increase the accuracy of the estimation and make the system more reliable.

For testing and verification of MASIS, the system was installed in the reactor in February 2006 and has been working since then. The performance of the system has been quite satisfactory until now. At this point, there has been no request for additional features from the operators. However, if, in future, operators need to change or add some extra features that they have not noticed until now, this is feasible with no complicated work as the source code is accessible and its detail is fully described in the thesis. In such a case, it would be shrewd to apply changes to a parallel system and keep the current running system unaffected. This would make it possible to compare the results of the unchanged system with those of the changed system to make sure that the later system is working accurately.

Checking for the reliability of the current system, it is believed that the calculations and the software of the MASIS are quite reliable as there are possibly few or no unpredictable parameters involved that have not been considered. However, the chance of an unpredicted problem in the hardware that may result in an accident is not practically zero. As a recommendation for future development to increase the reliability of the system, it is suggested to have two parallel hardware running in the MASIS. These parallel systems send the signals to the software where a comparison of the received signals decides on the accuracy of the data as well as the fidelity of the hardware; i.e. if there is a big difference between the two received signals, it indicates an error in the system. There is only some slight additions needed in the software to manage the parallel system; for example, two LabJacks must be introduced by adding their serial numbers in "IDNum" input of the count function. One example to show the necessity of having a parallel system is when the counter of the LabJack due to some problems counts less or

more that the real count-rate. If the counter could not work at all, the system shows the error in the screen. But if it counts inaccurately, there is no way to show such a fault in the system as there is no other value to compare the result with.

Appendix 1

Iodine

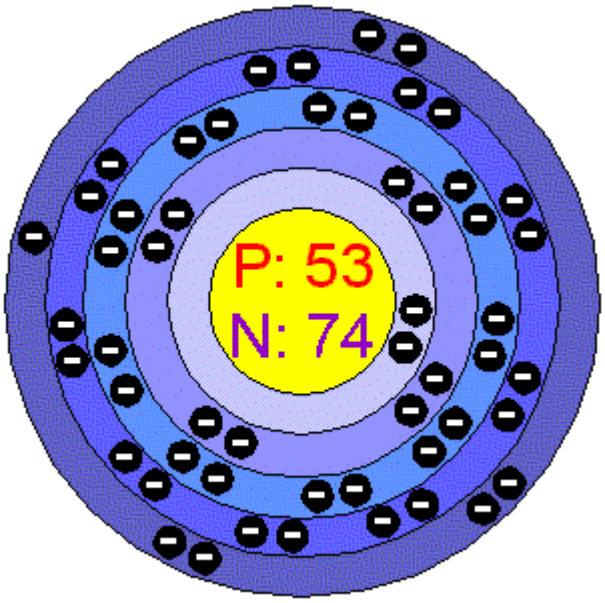
Iodine with symbol "I" has the atomic number of 53 and atomic mass of 126.90447 amu. The melting point of this blackish halogen is 113.5 °C (386.65 °K, 236.3 °F) and the boiling point is 184.0 °C (457.15 °K, 363.2 °F). Its crystal structure is orthorhombic and its density in 293 ° K is 4.93 g/cm$^{3.}$

Table 1 shows the different isotopes of iodine with the corresponding half life. The atomic structure of iodine is shown in Fig. I.

Isotope	**Half Life**
I-122	3.6 minutes
I-123	13.2 hours
I-124	4.2 days
I-125	60.1 days
I-126	13.0 days
I-127	Stable
I-128	25.0 minutes
I-129	1.57E7 years
I-130	12.4 hours
I-131	8.0 days
I-132	2.3 hours
I-133	20.8 hours
I-134	52.6 minutes
I-135	6.6 hours
I-136	1.4 minutes

Table 1: The isotopes of iodine and their half life time. [9]

.



Number of Energy Levels: 5

First Energy Level: 2
Second Energy Level: 8
Third Energy Level: 18
Fourth Energy Level: 18
Fifth Energy Level: 7

Fig. I: The atomic structure of iodine. [9]

Appendix 2

Iodine-125

Physical data:

Nuclear Reaction: $^{124}Xe \xrightarrow{(n.\gamma)} {}^{125}Xe \xrightarrow[17h]{(\beta,\gamma)} {}^{125}I$

Physical half life: 60.28 days

Mode of Decay: Electron capture

Decay product: Te^{125}

Principal Emissions:[10]

Gamma: 0.035 MeV (6.5%)

Kα X-ray: 0.027 MeV (112.5%)

Kβ X-ray: 0.031 MeV (25.4%)

Shielding:[11]

Half-Value Layer for Lead Shielding: 0.02 mm (0.001 in.)

Unshielded Exposure Rate for 1 mCi Point Source at 1 cm: 1.4 R/h

Unshielded Exposure Rate from 1 MBq Point Source at 1 m: 0.98 nC/kg/h

Biological Data:

Critical Organ: Thyroid

Radiotoxicity: high

Biological half-life: 138 days

Effective half-life: 42 days

Routes of Intake: Ingestion, inhalation (most probable), puncture, wound, skin contamination (absorption)

"It may be assumed that 30% of an uptake of iodine is translocated to the thyroid and 70% directly excreted in urine" [12]

Occupational Limits:[13]

Annual Limit on Intake: 1 MBq.

Derived Air Concentration: 600 Bq/m3.

Decay table of I-125 :

		Days									
		0	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**
Days	**0**	1.000	0.977	0.955	0.933	0.912	0.891	0.871	0.851	0.831	0.812
	20	0.794	0.776	0.758	0.741	0.724	0.707	0.691	0.675	0.660	0.645
	40	0.630	0.616	0.602	0.588	0.574	0.561	0.548	0.536	0.524	0.512
	60	0.500	0.489	0.477	0.467	0.456	0.445	0.435	0.425	0.416	0.406
	80	0.397	0.388	0.379	0.370	0.362	0.354	0.345	0.338	0.330	0.322
	100	0.315	0.308	0.301	0.294	0.287	0.281	0.274	0.268	0.262	0.256
	120	0.250	0.244	0.239	0.233	0.228	0.223	0.218	0.213	0.208	0.203
	140	0.198	0.194	0.189	0.185	0.181	0.177	0.173	0.169	0.165	0.161
	160	0.157	0.154	0.150	0.147	0.144	0.140	0.137	0.134	0.131	0.128
	180	0.125	0.122	0.119	0.117	0.114	0.111	0.109	0.106	0.104	0.102
	200	0.099	0.097	0.095	0.093	0.090	0.088	0.086	0.084	0.082	0.081
	220	0.079	0.077	0.075	0.073	0.072	0.070	0.069	0.067	0.065	0.064
	240	0.063	0.061	0.060	0.058	0.057	0.056	0.054	0.053	0.052	0.051

Table 2: The number of days is shown in the top and left hand column, and the corresponding decay factor is shown in the chart.

Appendix 3

NaI

Main Properties of NaI:[14]

Density	3.67 g/cm^3
Lattice constant	6.46 Å
Melting Point	651°C
Hardness	2 Mohs
Hydroscopicity	little
Decay constant	Fast620 ns Slow250 ns
Radiation length	2.6 cm
Emission wavelength	Fast310 nm Slow410 nm
Light yield	52 - 56×10^3 photons/MeV
Refractive Index (Maximum emission wavelength)	1.85
Linear attenuation coefficient	(for 511 kev γ-rays) 0.34 cm^{-1}
Light yield	Fast 6.5×10^3 Slow4×10^4 photon/Mev
Energy resolution(for 511)	about 8 Kev%
Flatness	λ/4 @ 633nm
Parallelism	<10"
Perpendicularity	<10'
Time resolution (for 60Co)	1000 ps

Appendix 4

LabJack U12[15]

Besides the DB25 channels at the top edge of the lab Jack and the CNT port at the top surface of the LabJack explained in the text, there are some other ports available as screw terminals at the top surface of the LabJack U12 that are shown in Fig. II

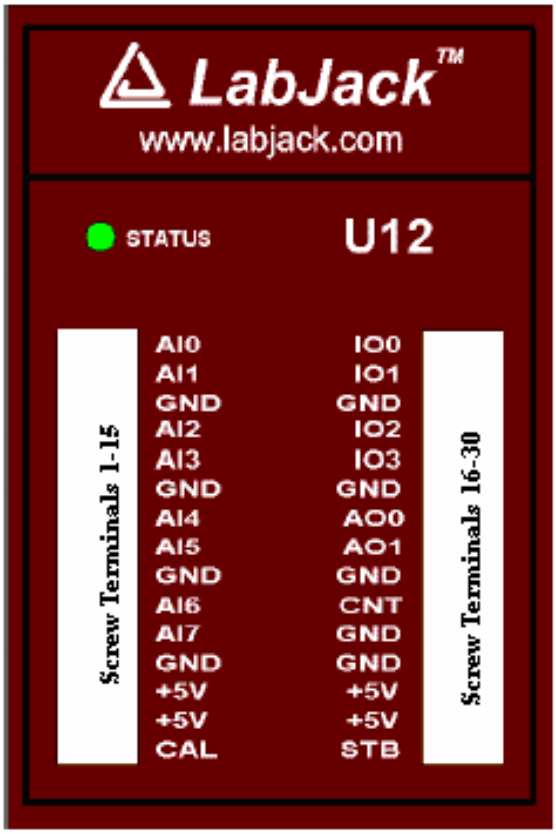


Fig. II: LabJack U12 top surface

AI0 –AI 7

AI0 to AI7 are eight analog inputs. The resolution of each input is 12-bit and the input bias current is 90 µA. These screw terminals can be configured individually and on-

the-fly as eight single-ended channels, four differential channels, or combinations in between.

AO0-AO1

These two screw terminals used for analog output voltages. AO0 and AO1 ports have 10-bits resolution. Each of these two can be set to a voltage between 0 and the supply voltage that is normally 5 volts.

IO0-IO3

LabJack U12 has twenty digital I/O ports. Sixteen of them are within the DB25 connector at the top edge of the LabJack. IO0 to IO3 are the other four digital I/O ports located at the screw terminals at the top of the surface of the LabJack. In order to protect the ports from overvoltage\short circuit, they are internally connected to a 1.5 $k\Omega$ series resistor. However, this protection reduces the current capability of the channels.

Each of these four channels can be set individually to input, output high, or output low state.

CAL-STB

Both of these two screw terminals are employed for testing and calibration. CAL can be used in normal operation. It is a precision 2.5 volt reference, but has a current limit of 1 mA. It also has a protection for ESD and overvoltage. However, severe overvoltage in steady state or transient can damages the CAL pin. This damage causes the failure of all analog inputs (AI0-AI7).

GND

Both screw terminals and DB25 connector have a number of GND connections. These GNDs are all alike (i.e. common ground).

+5V

There is an internal power supply in the LabJack with the nominal voltage of +5V. This voltage terminal is accessible through the +5V screw terminals or the +5V pins on the DB25 connector.

For most desktop PC and self powered USB hubs, 450 mA is the total amount of current that can be drawn from AOs (analog outputs), DOs (digital outputs) and +5V channels. This current can be limited to 50 mA by some kind of laptops or bus-powered hubs.

It is required to have overcurrent protection on all hosts and hubs using USB connection. In case of putting too large load on +5V pins of LabJack U12 or any other USB device, the current will be limited by the host or hub.

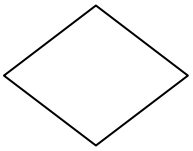
Appendix 5

Brief description of special symbols used in the flowcharts

There are special shapes and symbols in the flowcharts to represent different types of actions or steps in a procedure. A brief description of the shapes used in the flowcharts of the thesis is as follows:

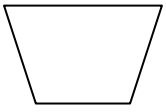
Terminator: This symbol usually contains the word "Start" or "End" and represents the starting or ending point of the flowchart

Process: This "Process" or Action box indicates a single step or a sub-process in the procedure of a flowchart.



Decision: This shape is the symbol of "Decision" or branching point. The lines/arrows pointing out from the diamond represent different choices.

Predefined Process: This "Predefined Process" or "Subroutine" box contains a sequence of actions that is usually described in a separate flowchart. This flowchart performs a specific assignment embedded within a larger process in the main flowchart.



Manual Operation: The tasks that operates manually are set in this box.

Appendix 6

Linear Fit PtByPt

"Linear Fit Coefficients PtByPt" is a sub-VI that is usually used to find the slope of the best linear fit with the method of "Least Square Solution". The appearance of this sub-VI is shown in Fig. III.

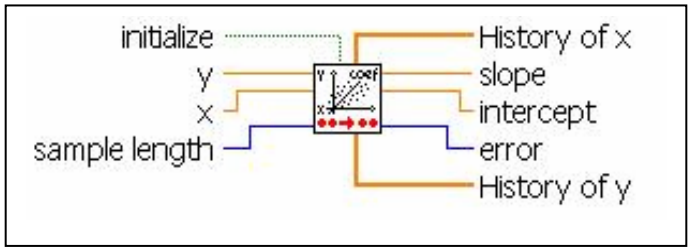


Fig. III: "Linear Fit Coefficients PtByPt".

As the program needs the value of Mean Square Error of the estimation to finding the Iodine Concentration Error, to avoid having an extra function in the main code, instead of "Linear Fit Coefficients PtByPt", "Linear Fit PtByPt" is employed. Fig. IV shows the appearance of "Linear Fit PtByPt".

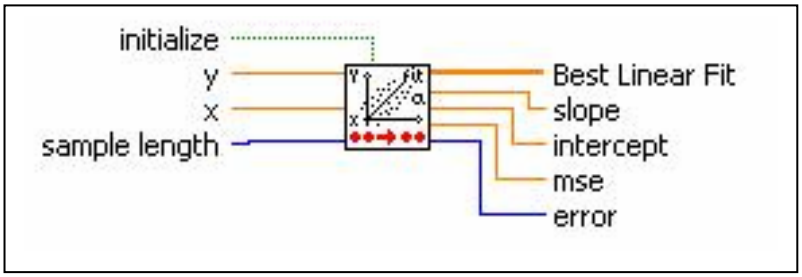


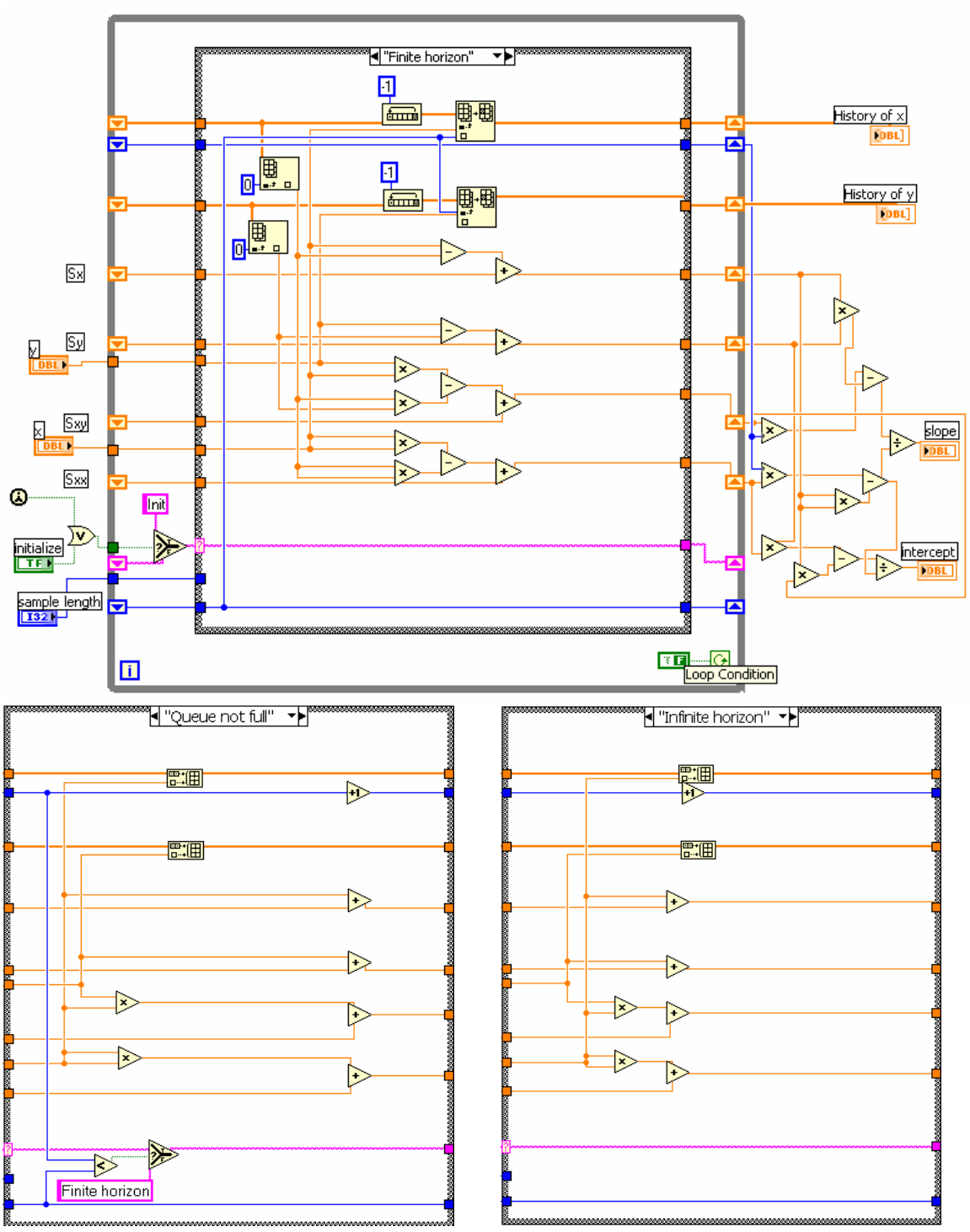
Fig. IV: "Linear Fit PtByPt".

The approach of "Linear Fit Coefficients PtByPt" to find the intended value is shown in Fig. V.

The relation between these two sub-VIs, "Linear Fit Coefficients PtByPt" and "Linear Fit PtByPt", is simple and is shown in Fig. VI.

In the discrete-time signal model, $y(n) = mn + b$, $y(n)$ is the "count-rate" and n is the "time". Therefore, in the main code, as seen in Fig. VII, "y" input of the function of "Linear Fit PtByPt" is connected to the "Count Rate (CPS)". "x" input of "Linear Fit PtByPt" is wired to a function that gives us the current time in second. The function of "Time in Second" is shown in Fig. VII by an arrow. This function is used for different tasks at many places in the main code. It is important that these tasks be synchronized; therefore, the program does not employ a separate "Time in Second" function for each assignment. The time that is wired to the input of x is given from a function that is not near the function of "Linear Fit PtByPt". Therefore, it cannot be shown in the figure. Equally, a local variable could have been used at this point.

To use the Least Squares Solution method, the program needs a set of incoming data. The length of each set is adjusted by the input of "sample length". Although the sample length is adjustable by the operator, the default value is 3600 samples. Since the iteration time in the program is about 1 second, in the case of having 3600 samples, the initializing period for the program is about one hour. A larger sample length results in a more accurate result. However, accuracy of 3600 samples is adequate enough. In the initializing period, the program does not have an accurate value for the "Slope of Count-rate in Time" and thus for the Iodine Concentration. After this period, every second the program has a precise estimation based on the last 3600 samples.


"Finite horizon"
History of x
History of y
Sx
Sy
Sxy
Sxx
Init
initialize
sample length
slope
intercept
Loop Condition
"Queue not full"
Finite horizon
"Infinite horizon"

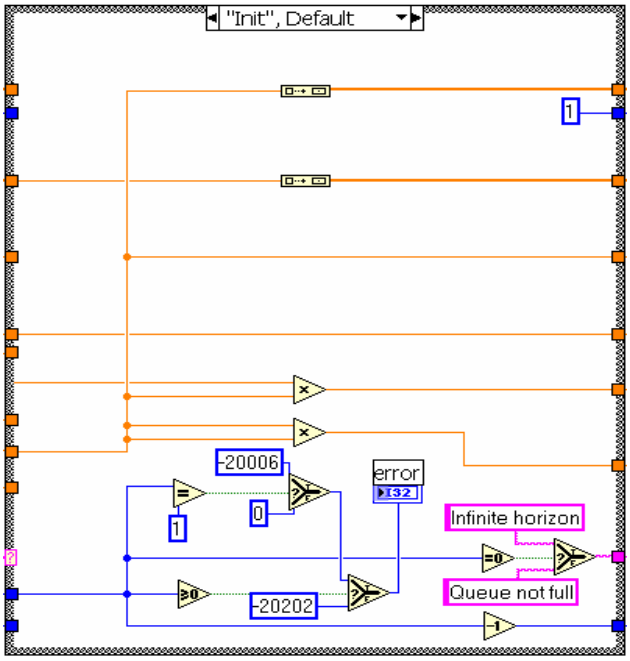


Fig. V: "Linear Fit Coefficients PtByPt". This code shows how Least Square Solution is employed to find the slope and intercept of the best linear fit. The sequence of the sub-diagrams in the Stacked Sequence Structure is: 1) "Queue not filled" 2) "Finite horizon" 3) "Infinite horizon" 4) "Init, Default".

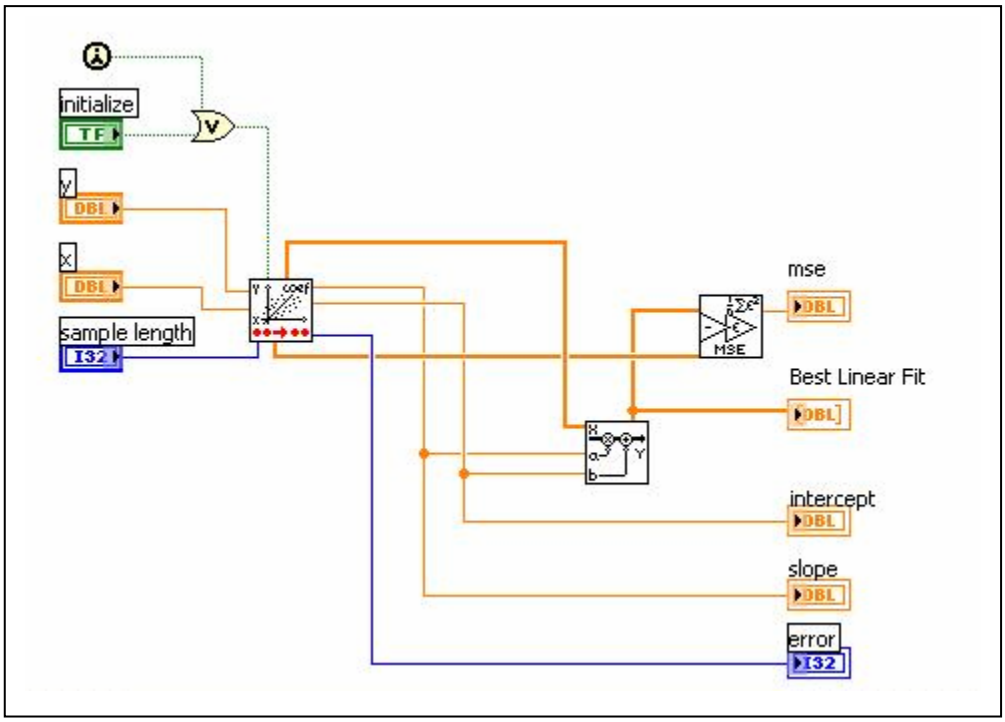


Fig. VI: "Linear Fit PtByPt". This code shows the relation between "Linear Fit Coefficients PtByPt" and "Linear Fit PtByPt".

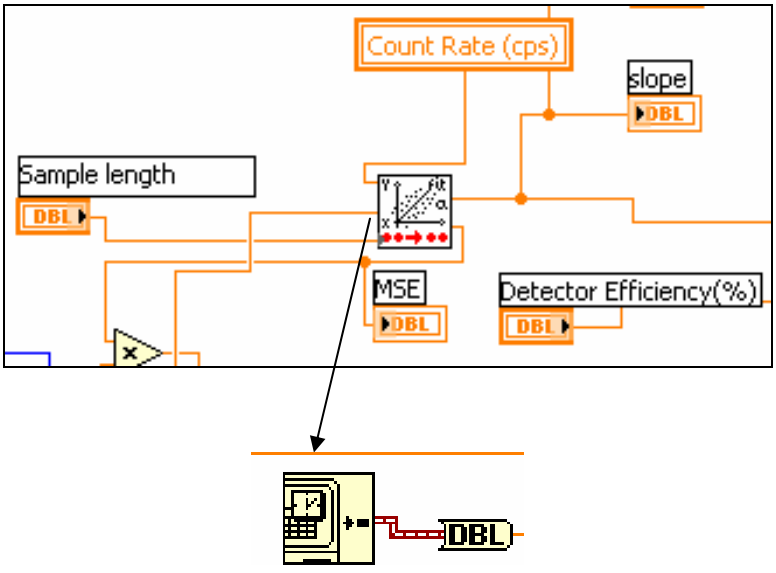


Fig. VII: "Linear Fit PtByPt" in the main code.

Appendix 7

"EDigitalOut"

Fig. VIII shows the code of the sub-VI "EDigitalOut"

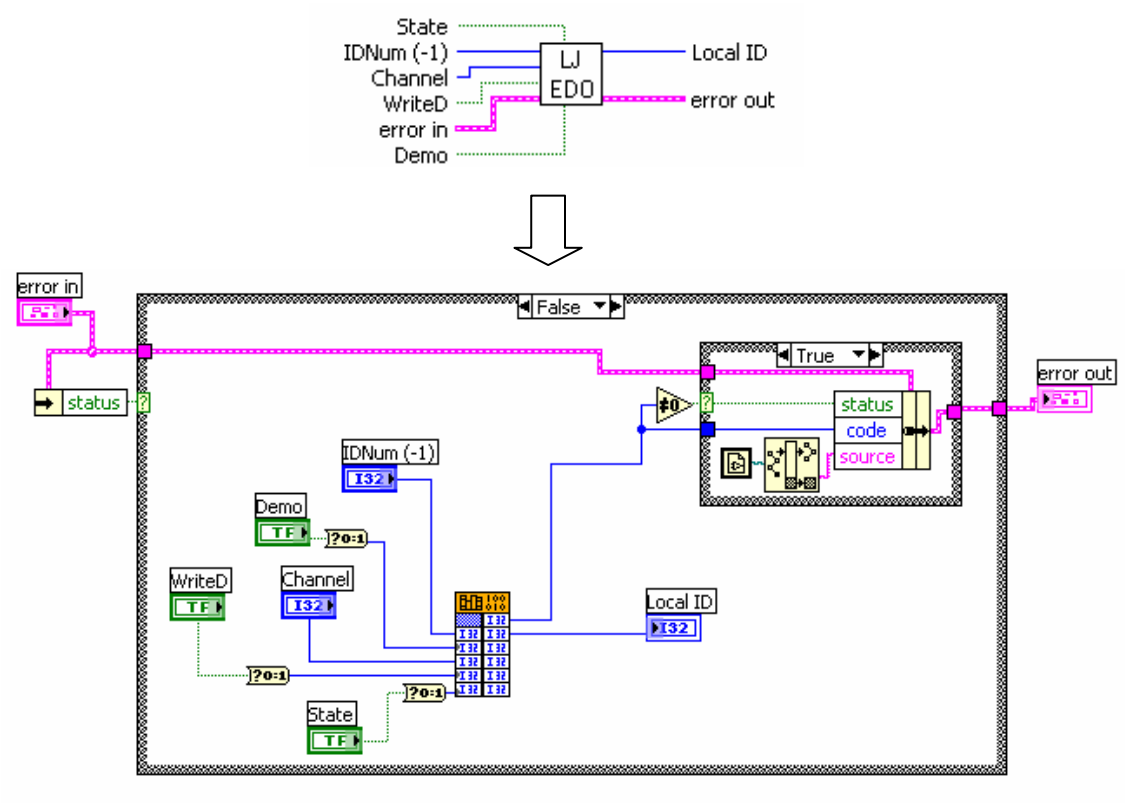


Fig. VIII : The code of "EDigitalOut". The other cases of the both case structures contain nothing only connections between "error in" and "error out".

In the following, some input and output parameters of the function of "EDigitalOut" are described.

Inputs:

-State: The line is cleared if State≤0 and set if State>0. It also can work by Boolean constants.

-IDNum(-1): The default is "-1" which refers to the first found LabJack. This input can remain the default value when there is only one LabJack in the system. If there are several LabJacks in the system, the local ID or serial number of the desired LabJack should be connected to "IDNum(-1)".

-Channel: It is the number of the pin. It can be 0 to 3 for screw terminal I/O or 0 to 15 for D lines.

-WriteD: If WriteD is grater than zero, D lines are supposed to be taken, otherwise screw terminal I/O will be written. Also True/False Boolean can be wired to this input.

-Demo: For normal operation it is "0". Any number greater than it means the program is in demo state. Similar to the other inputs, Boolean constants can be used instead of numbers. In Demo state, this function works without any LabJack.

Outputs:

-Local ID: This output shows the input IDNum. When No LabJack is found, Local ID is "-1" again.

Appendix 8

XY Chart Buffer

"XY Chart Buffer" is a sub-VI that is used when transferring data into a graph is required. Fig. IX shows the appearance of this sub-VI. As a matter of fact, this sub-VI mimics the behavior of the XY graph. With the help of this sub-VI, it is possible to make many calls to the VI without confusing the memory.

When the program needs to supply just one point at a time, it uses the input of "new point" in the "XY chart Buffer" function. This input is a cluster. For our purpose in both graphs of MASIS, the "new point" input is suitable. The input of "new points" is used when supplying multiple points at once is needed. This input is an array.

The "chart length (points)" is the length of the most recently data points to be demonstrated in the graph. We can not have access to the data in excess of the chart length as that is discarded.

"clear first" can be set to clear the history of graph before processing the new data points.

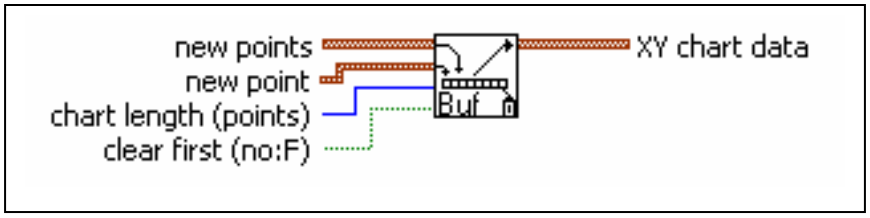


Fig. IX: The appearance of "XY chart Buffer".

The code of this sub-VI is shown in Fig. X that is the block diagram of the VI. Fig. XI shows its Front Panel. As it is seen in these figures the inputs of this VI are just like the inputs of function "XY chart Buffer" and included "new points" or "new point", "chart length" and "clear first". The output, XY chart data, is also similar.

If there is no access to the sub-VI of "XY chart Buffer", that is likely to happen in some LabVIEW packages, this code can be used to build a buffer for the graph.

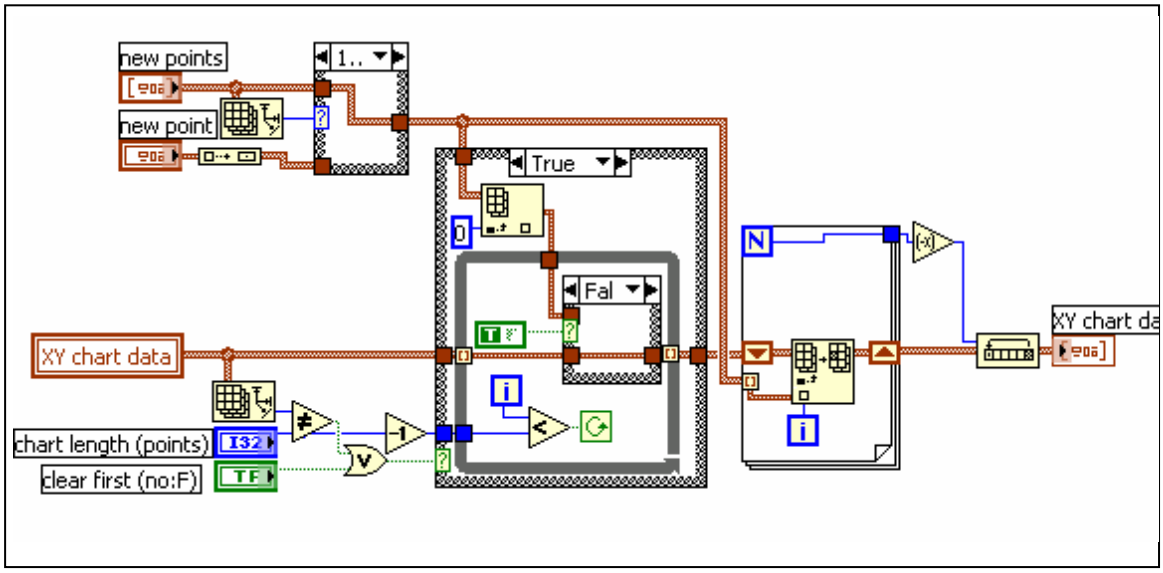


Fig. X: The code of sub-VI "XY chart Buffer".

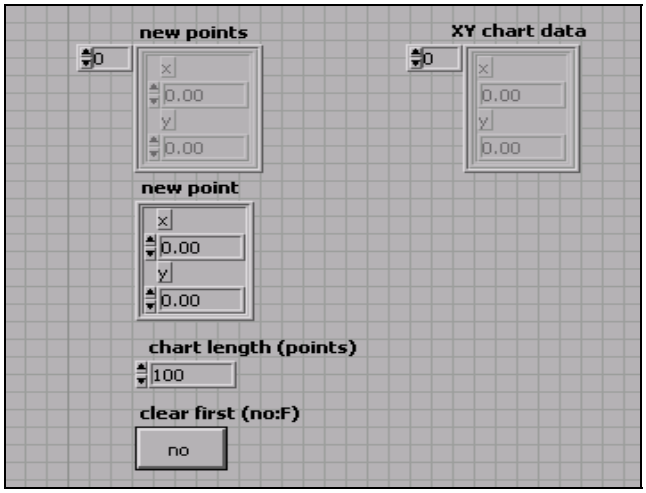


Fig. XI: The related front panel of "XY chart Buffer".

References

[1] http://www.mds.nordion.com/master_dex.asp?page_id=1019

[2] Razvan Simionescu, Iodine monitoring and advising system, (Hamilton, McMaster University) M.Sc. Dissertation, 2000.

[3] Kay, Steven M., Fundamentals Of Statistical Signal Processing: Estimation theory, Volume 1, Prentice-Hall PTR, Upper Saddle River, N.J., 1993.

[4] Niven Erin, Calibration of health physics instrument, (Hamilton, McMaster University) M.Sc. Dissertation, 1998.

[5] http://www.automation.com

[6] Signal Tower manual www.patlite.com

[7] TiacOut4 manual. www.SimpleIO.com

[8] Information about LabVIEW can be found at National Instrument's website: http://www.ni.com

[9] Bentor, Yinon. Chemical Element.com - Iodine. Jan. 27, 2006. http://www.chemicalelements.com/elements/i.html

[10] 1. Kocher, David C., Radioactive Decay Data Tables, Springfield: National Technical Information Service, 1981 DOE/TIC-11026.

[11] Calculated with computer code "Gamma" utilizing decay scheme data from Kocher and mass attenuation coefficients for lead and mass energy absorption coefficients for air from the Radiological Health Handbook, Washington: Bureau of Radiological Health, 1970. (The HVL reported here is the initial HVL for narrow beam geometry).

[12] ICRP Publication 30, Part 2, Limits for Intakes of Radionuclides by Workers. Pergamon Press, Oxford, 1979.

[13] ICRP Publication 68, Dose Coefficients for Intakes of Radionuclides by Workers, Pergamon Press, Oxford, 1995.

[14] http://www.optical-components.com

[15] Lab Jack U12 users guide.