MAUSAM, **67**, 2 (April 2016), 499-504

551.501.75 : 551.508

# PC based new software developed to create an input pilot balloon data file to an alternative to Hand Held Data Logger (HHDL) for using PC based SAMEER Pibal computation software

N. MEENATCHI NATHAN, CHANABASANAGOUDA. S. PATIL and J. P. IMMANUEL JAYAPRAKASH

*Aerodrome Meteorological Office, Chennai, India*

(*Received 3 October 2013, Accepted 7 September 2015*)

**e mail : nathan.imd@gmail.com**

**सार** – वर्ष 2007 से उपरितन वायु आँकड़ों का आकलन करने के लिए SAMEER द्वारा निर्मित हैंड हेल्ड डेटा लॉगर (HHDL) का उपयोग भारत मौसम विज्ञान विभाग द्वारा पवन सूचक गुब्बारा प्रेक्षणों के लिए किया जा रहा है। HHDL एक पतला और सूक्ष्म नियंत्रक आधारित बैटरी चालित घटक है जो अपरिष्कृत फाइल तैयार करने और पवन सूचक गुब्बारा आँकड़ा संसाधन के लिए PB आरोहण के अनुसार संख्यात्मक पैड के माध्यम से सभी सूचनायें एकत्रित करता है। अपरिष्कृत फाइल PC आधारित पाइबल आकलन सॉफ्टवेयर को सूचना फाईल के रूप में कम्प्यूटर प्रणाली को भेजता है। यह सॉफ्टवेयर राष्ट्रीय आँकड़ा केंद्र (NDC) डेटा फार्मेट और मासिक जलवायु के अतिरिक्त HHDL के समरूपी पाइबल संदेश तैयार करता है। हॉर्डवेयर में किसी प्रकार की विफलता आने पर दोनों HHDL और PC आधारित पाइबल आकलन सॉफ्टवेयर का उपयोग नहीं किया जा सकता है। अत: इस समस्या से निपटने के लिए दृश्य C शार्प का उपयोग करते हुए PC आधारित पाइबल आँकड़ा सॉफ्टवेयर विकसित किया गया। विकसित किया गया नया सॉफ्टवेयर HHDL के समान ही सूचना फाइल बनाता है, इसकी जाँच PC आधारित पाइबल आकलन सॉफ्टवेयर के साथ की गई है जो HHDL और इसके हार्डवेयर के सहायक उपकरणों की विफलता के समय विकल्प के रूप में सफलतापूर्वक कार्य करता है।

**ABSTRACT.** Pilot balloon observatories of India Meteorological Department (IMD) are using Hand Held Data Logger (HHDL), manufactured by SAMEER, to compute upper air data since 2007. The HHDL, which is a sleek and microcontroller based battery operated unit, accepts all information through the numeric keypad pertaining to the PB ascent for raw file generation and pilot balloon data processing. The raw file can be transferred to computer system as an input file to PC based Pibal computation software. This software generates Pibal messages similar to HHDL in addition to National Data Centre (NDC) data format and monthly climate. In case of any failure of hardware, both HHDL & PC based Pibal computation software cannot be used. Therefore to overcome this problem, a PC based Pibal data keying software has been developed using visual C sharp. The new software, what is developed, creates an input file similar to HHDL; it was tested with PC based Pibal computation software which works successfully as an alternate in case of failure of HHDL & it’s hardware accessories

**Key words** – Azimuth and elevation angle information, Input file generation, HHDL-Hand Held Data Logger, Pibal-pilot balloon, Theodolite, Visual C sharp.

## 1. Introduction

The pilot balloon observations, whose observations dates back to early 20th century, are still used today in mostly developing countries in Asia and Africa. Although PB observations are having limitations due to cloudiness, their low cost has historically allowed for more frequent and widespread used than radio sonde.

Various instruments and methods are used to measure wind, an inevitable parameter in meteorology and forecasting weather, at surface and at upper air level. Pilot balloon observation is one of the techniques of upper air wind measurement. A spherical rubber balloon, filled with hydrogen gas, rises due to its buoyancy forces and gets drifted by wind when it is released. The balloon is followed with the help of optical theodolite and its horizontal and vertical angles are noted with the graduated scale fixed on the theodolite. At the interval of every minute the height of the balloon above ground is calculated either assuming fixed rate of ascent or with the help of tail attached to the balloon. The angular position of the balloon is determined in terms of two quantities *viz*, azimuth and angle of elevation. The azimuth angle is considered horizontally from the true north and the angle of elevation measures the position in the upward direction. The azimuth angle is denoted by H and the angle of elevation is by 'V'. In the field of view of the telescope of the theodolite the flag readings are noted which are known as graticule reading. With the help of these three parameters the computational work is done to get the height drift of the balloon at every interval manually by drawing the trajectory and using suitable velocity scale for measuring the wind velocity.

In 2007 under modernization programme of IMD, all the PB stations are supplied with micro controller based Hand Held data logger. The Hand Held Data Logger (HHDL) is a sleek, microcontroller based battery operated unit for processing Pilot balloon data. Using Pilot balloon flight data, the HHDL computes wind speed and direction at 1-minute interval, at standard height and standard pressure levels. At the end, it also generates the Pilot message. The operator can print the processed data in the tabular form on the serial printer.

It has built-in battery backed memory to archive 100 numbers of the flight data. HHDL Accessories are RJ-9 & RJ-11 Coiled cable, RJ to D type 9 pin connecter PC cable, Printer cable, Battery charger, three numbers of rechargeable batteries, etc. The theodolite observed parameters are keyed to the HHDL through Alpha numeric keypad which contains total 24 keys which are used for entering alphanumeric values. Various combinations of these keys are used for entering data, processing, connecting to PC etc. Hence these operations are time consuming and needs a good amount of practice for using the data logger. Moreover, these accessories and data logger is getting old, gives frequent problems and become unserviceable in some stations. SAMEER has supplied a Pibal Data Processing software (PC) to process the HHDL raw data files after transferring in to PC. This PC based software is capable of generating Pibal code, NDC data archival format and monthly climate etc. If the HHDL goes unserviceable then (PC based data processing software) this cannot be used and the total Pibal computation has to revert back to Manual computations.

These drawbacks may be overcome by using newly developed PC based software for pilot data inputting using visual studio built in language visual C#. Visual Studio is an Integrated Development Environment (IDE) from Microsoft. It is used to develop console and graphical user interface along with windows form applications etc. The Visual Studio built-in tools include a forms designer for building Graphical User Interface (GUI) applications, web designer, class designer and database schema designer etc. Visual Studio includes a full text generation framework called T4 which enables Visual Studio to generate text files from templates either in the IDE or *via* code. Using these features GUI based PB data input forms created and a suitable text file has been created for using the PC based PB computation software supplied by SAMEER.

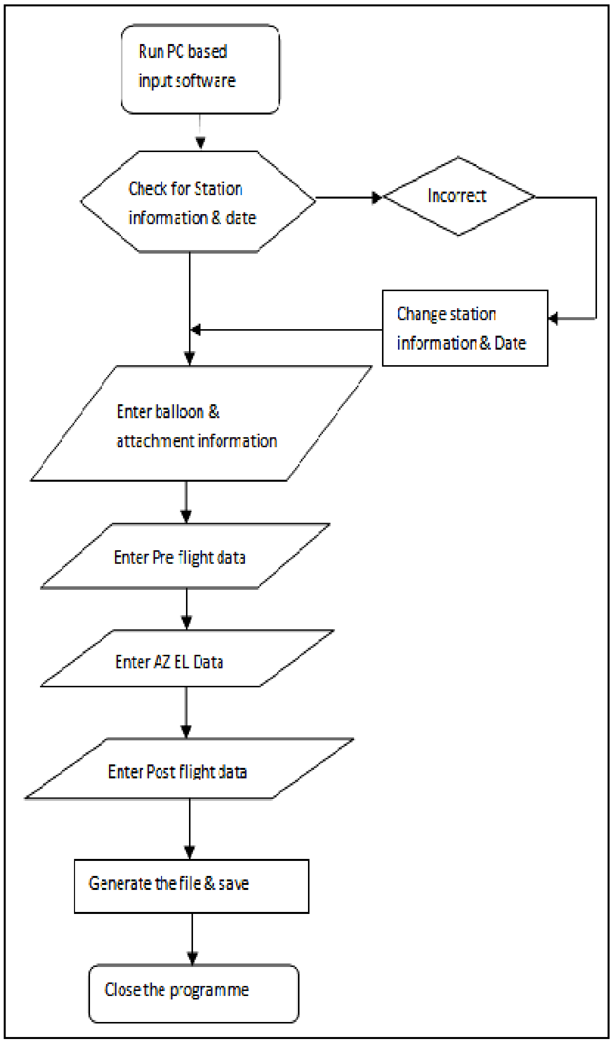
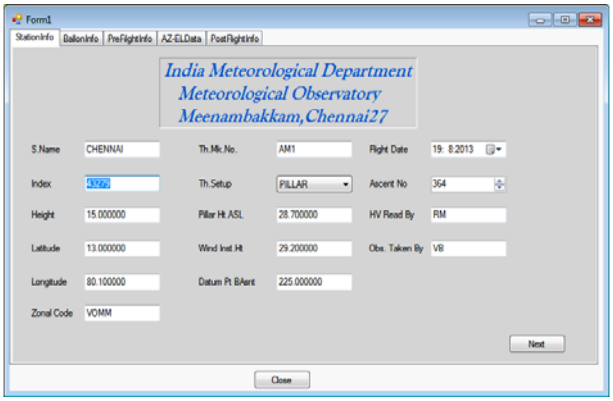

Run PC based input software
Check for Station information & date
Incorrect
Change station information & Date
Enter balloon & attachment information
Enter Pre flight data
Enter AZ EL Data
Enter Post flight data
Generate the file & save
Close the programme

**Fig. 1.** Pilot balloon – PC based data entry hierarchy flow chart

## 2. Data and methodology

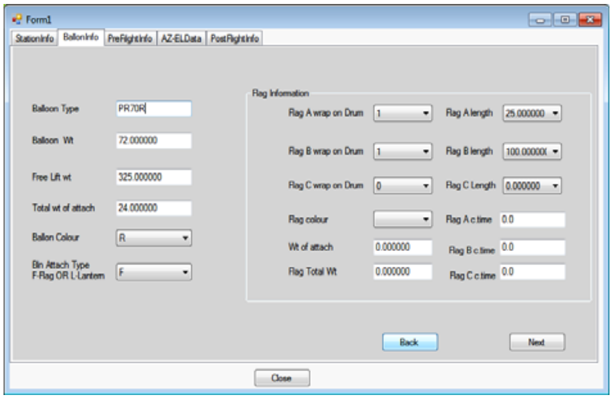
Pibal computation needs various information such as station details, balloon and attachments, preflight, post flight cloud information and theodolite angle information's of the flight at every interval of one minute. The visual studio GUI tool consists of various options of text box, combo box, date time picker, Label, and so on. Different information collected through a tab control



*India Meteorological Department*
*Meteorological Observatory*
*Meenambakkam,Chennai27*

| | | | | | |
|---|---|---|---|---|---|
| S.Name | CHENNAI | Th.Mk.No. | AM1 | Flight Date | 19: 8:2013 |
| Index | 43279 | Th.Setup | PILLAR | Ascent No | 364 |
| Height | 15.000000 | Pillar Ht.ASL | 28.700000 | HV Read By | RM |
| Latitude | 13.000000 | Wind Inst.Ht | 29.200000 | Obs. Taken By | VB |
| Longitude | 80.100000 | Datum Pt BAsnt | 225.000000 | | |
| Zonal Code | VOMM | | | | |

**Fig. 2.** Station information window



| | |
|---|---|
| Balloon Type | PR70R |
| Balloon Wt | 72.000000 |
| Free Lift wt | 325.000000 |
| Total wt of attach | 24.000000 |
| Ballon Colour | R |
| Bln Attach Type<br>F-Flag OR L-Lantern | F |

Flag Information

| | | | |
|---|---|---|---|
| Flag A wrap on Drum | 1 | Flag A length | 25.000000 |
| Flag B wrap on Drum | 1 | Flag B length | 100.00000( |
| Flag C wrap on Drum | 0 | Flag C Length | 0.000000 |
| Flag colour | | Flag A c.time | 0.0 |
| Wt of attach | 0.000000 | Flag B c.time | 0.0 |
| Flag Total Wt | 0.000000 | Flag C c.time | 0.0 |

**Fig. 3.** Balloon & attachment window

windows form application, which gives easy access to moves around. The hierarchy of data feeding explained in the flowchart.

### 2 (a). *Station information*

The station information consists of basic details of the station which is used to identify the station as well as wind computation. The station information gives default values for specific stations, correction if needed; it permits the correction and formats the inputs automatically. The sample programme explained for pillar height. The Pillar height, being a floating point number, has to accept only the numbers with one decimal point  and not an alphabet. Hence, the key press event & mouse leave event are programmed, as follows, for accepting the inputs and automatically converts to the specified formats.

```
private   void   PillarHtASL_KeyPress(object    b
sender, KeyPressEventArgs e)
       {
           // string oldText = string.Empty;
            if (!char.IsControl(e.KeyChar)
                  && !char.IsDigit(e.KeyChar)
                  && e.KeyChar != '.')
            {
                e.Handled = true;
            }

            // only allow one decimal point
            if (e.KeyChar == '.'
                && (sender as TextBox).Text.
                IndexOf('.') > -1)
            {
                e.Handled = true;
            }
        }
```

```
        private void PillarHtASL_Leave(object
        sender, EventArgs e)
        {
            PillarHtASL.Text= String.Format
            ("{0:0.000000}",
            double.Parse(PillarHtASL.Text));
        }
```

### 2. (b). *Balloon and attachments*

The balloon and type of attachment information is important for deciding the fixed rate of ascent for night and tail attachments for day time wind calculation. This window provides an option to select the type of attachment as lantern for night and flag for data time. If flag is selected, other relevant information pertaining to the flag such as  length of the different flags, drum number of the flags, flag change over time  for graticule measurements are needed. Depending on the type of attachments, different data grid view created for day and night observations. The coding is as follows :

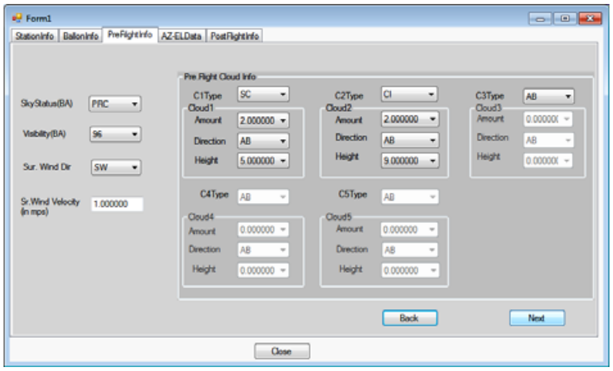
```
private void BalloonAttachType_SelectedIndex
Changed(object sender, EventArgs e)
        {
            if (BalloonAttachType.Text.
            Contains("F"))
            {
                FlagInformation.Enabled = true;
                dataGridView2.Enabled = true;
                dataGridView2.Visible = true;
                dataGridView1.Enabled = false;
                dataGridView1.Visible = false;
            }
            else
            {
               FlagInformation.Enabled = false;
               dataGridView2.Enabled = false;
```

```
          dataGridView2.Visible = false;
          dataGridView1.Enabled = true;
          dataGridView1.Visible = true;
        }
```

### 2. (c). *Cloud information*

The preflight and postflight window gives the surface wind, visibility and various cloud information before and after the ascent. This is achieved using combo box; it displays an editable text box with a dropdown list of permitted values. If the sky condition is partly cloudy then only it permits to enter the other cloud information otherwise totally disabled the cloud information. The following is the sample coding and Preflight window.

```
private void SkyStausBA_SelectedIndexChanged(object
sender, EventArgs e)
    {
      if (SkyStausBA.Text.Contains("PRC"))
      {
        PreFlightCloudInfo.Enabled = true;
      }
      else
      {
        PreFlightCloudInfo.Enabled = false;
        PrC1Type.SelectedIndex = 0;
        PrC1Amt.SelectedIndex = 0;
        PrC1Dir.SelectedIndex = 0;
        PrC1Ht.SelectedIndex = 0;
      }
    }
private void PoC1Type_SelectedIndexChanged(object
sender, EventArgs e)
    {
      if (PoC1Type.Text.Contains("AB"))
      {
        PoCloud1.Enabled = false;
        PoCloud2.Enabled = false;
        PoCloud3.Enabled = false;
        PoCloud4.Enabled = false;
        PoCloud5.Enabled = false;
        PoC2Type.Enabled = false;
        PoC3Type.Enabled = false;
        PoC4Type.Enabled = false;
        PoC5Type.Enabled = false;
      }
      else
      {
        PoCloud1.Enabled = true;
        PoCloud2.Enabled = false;
        PoCloud3.Enabled = false;
        PoCloud4.Enabled = false;
        PoCloud5.Enabled = false;
       PoC2Type.Enabled = true;
        PoC3Type.Enabled = false;
        PoC4Type.Enabled = false;
        PoC5Type.Enabled = false;
      }
    }
```



SkyStatus(BA): PRC
Visibility(BA): 96
Sur. Wind Dir: SW
Sr.Wind Velocity (in mps): 1.000000

C1Type: SC
Cloud1
Amount: 2.000000
Direction: AB
Height: 5.000000

C2Type: CI
Cloud2
Amount: 2.000000
Direction: AB
Height: 9.000000

C3Type: AB
Cloud3
Amount: 0.00000(
Direction: AB
Height: 0.00000(

C4Type: AB
Cloud4
Amount: 0.000000
Direction: AB
Height: 0.000000

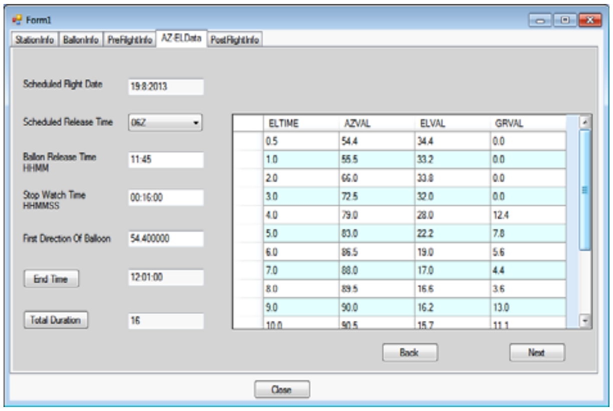
C5Type: AB
Cloud5
Amount: 0.000000
Direction: AB
Height: 0.000000

**Fig. 4.** Pre-flight cloud information window

### 2. (d). *Angle values of theodolite azimuth/elevation data*

On completion of the pre-ascent data entry, it is ready to accept balloon position data at one minute interval. The balloon position is defined in terms of graticule, azimuth and elevation angles for day time flight and no graticule values for night ascent (assuming uniform rate of ascent (9.0 km/hr)). This data is used to compute the wind speed and direction. This data entry is always starts from 0.5 minutes of elapsed time, and ready to accept Azimuth, Elevation and Graticule values in the data grid view window. The data grid view is configured to accept the angle values of the Pibal information's sequentially with auto increment of the Elapsed time with one minute interval using visual C# Event handler of the key press and row added event.

```
private void dataGridView2_KeyPress(object s1,
KeyEventArgs sd)
    {
      if (sd.KeyCode == Keys.Enter)
      {
        int c = this.dataGridView2.CurrentCell.
        ColumnIndex;
        int ct = this.dataGridView2.ColumnCount - 1;
        int r = this.dataGridView2.CurrentRow.Index;
        if (c == ct)
      {
        if (r >= 0)
        this.dataGridView2.CurrentCell =
        dataGridView2.Rows[r].Cells[0 + 1];
      }
```



Scheduled Flight Date: 19:8:2013

Scheduled Release Time: 06Z

Ballon Release Time HHMM: 11:45

Stop Watch Time HHMMSS: 00:16:00

First Direction Of Balloon: 54.400000

End Time: 12:01:00

Total Duration: 16

| ELTIME | AZVAL | ELVAL | GRVAL |
|---|---|---|---|
| 0.5 | 54.4 | 34.4 | 0.0 |
| 1.0 | 55.5 | 33.2 | 0.0 |
| 2.0 | 66.0 | 33.8 | 0.0 |
| 3.0 | 72.5 | 32.0 | 0.0 |
| 4.0 | 79.0 | 28.0 | 12.4 |
| 5.0 | 83.0 | 22.2 | 7.8 |
| 6.0 | 86.5 | 19.0 | 5.6 |
| 7.0 | 88.0 | 17.0 | 4.4 |
| 8.0 | 89.5 | 16.6 | 3.6 |
| 9.0 | 90.0 | 16.2 | 13.0 |
| 10.0 | 90.5 | 15.7 | 11.1 |

**Fig. 5.** Azimuth & elevation angle information window

```
        else
       {
        int r1 = r - 1;
        this.dataGridView2.CurrentCell=dataGridView2.
        Rows[r1].Cells[c + 1];
       }
       }
    }
private void dataGridView2_RowsAdded(object sender,
DataGridViewRowsAddedEventArgs e)
    {
       for (int i = 0; i < e.RowCount; i++)
       {
       if (e.RowIndex == 1)
       {
       this.dataGridView2.Rows[e.RowIndex -
       1].Cells[0].Value = (e.RowIndex - 1)  +
       ".5".ToString();
       }
     else
       {
       this.dataGridView2.Rows[e.RowIndex -
       1].Cells[0].Value = (e.RowIndex - 1) +
       ".0".ToString();
       }
       }
    }
```

2. (e). *File creation and saving*

Data collected through various windows needs to be saved in text document, using stream writer and save file dialog. The file location may be chosen at the time saving by the user. Sample coding for saving the file as follows.

```
private void Save_Click(object sender,
EventArgs e)
          {
              System.IO.StreamWriter objWriter;
              string saved_File = "";
              saveFD.InitialDirectory = "C:";
              saveFD.Title = "Save a Text File";
              saveFD.FileName = "";
              saveFD.Filter = "Text Files|*.PBL|
             Text Files|*.txt";

              if (saveFD.ShowDialog() ==
             DialogResult.Cancel)
              {
                  MessageBox.Show("Operation
                  Cancelled");
              }
              else
              {
                  saved_File = saveFD.FileName;
                  objWriter = new
                  System.IO.StreamWriter(saveFD.
                  FileName);
                   //Station Information

                  objWriter.WriteLine(SName.Text
                  );

                  objWriter.WriteLine(Index.Text
                  );

                  objWriter.WriteLine(SHeight.Te
                  xt);
               if
                  (BalloonAttachType.Text.Contai
                  ns("F"))
               {

objWriter.WriteLine(dataGridView2.Columns[0].
HeaderText + "\t" + dataGridView2.Columns[1].
HeaderText + "\t" + dataGridView2.Columns[2].
HeaderText + "\t" + dataGridView2.Columns[3].
HeaderText);
               int rowcount = dataGridView2.
               Rows.Count;
               for (int i = 0; i < rowcount - 1;
               i++)
               {

objWriter.WriteLine(dataGridView2.Rows[i].Cel
ls[0].Value.ToString() + "\t" + dataGrid
View2.Rows[i].Cells[1].Value.ToString() +
"\t" + dataGridView2.Rows[i].Cells[2].Value.
ToString()
                + "\t" + dataGridView2.
                 Rows[i].Cells[3].Value.ToString
                 ());
               }
          }

                objWriter.Close();
```

```
        MessageBox.Show("Data Successfully
Exported");
            }

        }
```

## 3. Result and discussion

Newly developed software has tested at various stages of the development and with various degrees of rigour for fulfilling the computation software formats. This software has successfully created an input file, and tested with SAMEER Pibal computation software. Day and night ascent real time data are tested for more than three months at Chennai. It is observed that it is very user friendly than the HHDL and successfully creating an input file. Copy of the software has given to M.C. Hyderabad (meteorological observatory) for their use since HHDL is out of service. This software will be a good standby/ alternate for the HHDL in case of failure. No limitations are found for archival/storage of the input files and processed files because it is stored directly in the system. This software is portable to any computer system with wind ows 7 or windows XP with visual studio updated versions or any higher versions of windows OS. Fig. 1 to Fig. 5 explains the steps to use this software. Simply run it for entering the Pibal input data and creating the file. This will solve the problem of reverting back to manual computation incase of failure of HHDL Hardware at all PB Observatories.

*Acknowledgment*

We are thankful to Dr. R. Suresh, Director in-charge of AMO Chennai for his helpful suggestions and encouragement in developing and testing the software. Also we thank all the staffs of observatory, Meenambakkam for extending their help for testing this software using the PB real time data’s.

### Reference

SAMEER make Pibal Hand Held Data Logger for IMD User manual

www.homeandlearn.co.uk/csharp/