INDUSTRIAL PLACEMENT LOGBOOK

Week 1: Monday 24/07/2000 to Friday 28/07/2000

Induction: -

- Company History.
- Company Rules.
- Company dress code.
- Employee Health and Safety Handbook.
- Quality Assurance: It is the quality policy of the company, in accordance with International Standards ISO 9001 to design and manufacture quality products and provide services to satisfy customer's needs and to meet all applicable statutory and regulatory requirements.

Factory Tour: -

- Main Office Block.
- CEC
- Electronic LAB
- Main factory – shop floor.

Was told that my first project would be software development, "Modbus communications for Access 4000", using turbo C 3.0 for dos, so spent some time getting familiar with turbo C and creating small programs. I have had experience with Turbo C before, so this was not a problem.

Detailed technical information about Modbus was obtained, and I familiarised myself with the protocol. Before I had never heard of the Modbus communication protocol.

Week 2: Monday 31/07/2000 to Friday 04/08/2000

The detailed technical specification for the Modbus Communication Protocol for Access 4000, was not completed yet, and would not be released until next week. So was given the task of designing a user friendly test program, to test the Modbus communication protocol, this test program was not specific to Access 4000, it was designed to be flexible and theoretically could be used to test any Modbus communication protocol, no matter the application.

Modbus test program: -

- Main Menu
    1. Wait for Response Frame:
    2. Build & Transmit Query:
    3. Build Query with bad CRC:
    4. Manual RTU frame with CRC:
    5. Manual RTU frame no CRC:
    6. Auto Random blocks – TX:
    7. Simulate Random Noise:
    8. Change Baud Rate
    9. About
    10. Quit
- Code needs to be readable, reusable, by others and I.
- Program designed, coded, tested, and checked by me.
- Changes will be made at a later date, making the test program more specific to the test of Modbus on Access 4000.
- Functions include: -

```
void main_menu(void);
void interrupt com_handler(void);
void trans(int c);
int  readchr (char *c);
void test_TX(void);
```

```
void test_RX(void);
unsigned short CRC16(unsigned char *message,unsigned short
data_length);
int  RD_function(int function, char *func_str);
void RD_error(int err, char *func_str);
int wait_chr(char *wc);
void test_CRC(void);
void manual_RTU(void);
void auto_RTU(void);
void rand_NOISE(void);
void manual(void);
void baud_rate(void);
void INPUT(unsigned short int *temp);
void about(void);
```

Week 3: Monday 07/08/2000 to Friday 11/08/2000

Received a copy of the Modbus communication protocol for access 4000 technical specification, which contained all the details required for designing the Modbus protocol for Access 4000. I was given Source-code form the exciting FG Wilson protocol, to use as a reference, to make sure the Modbus protocol will be compatible with the program structure of the access 4000 panel without major changes to exciting software (e.g. Modbus will be called every 125mS).

Summery: Modbus Protocol: -

- Modbus protocol is a messaging structure, widely used to establish master-slave communication between intelligent devices. A Modbus message sent from a master to a slave contains the address of the slave, the "command" (e.g. "read register" or "write register"), the data, and a check sum.
- Since Modbus protocol is just a messaging structure, it is independent of the underlying physical layer. It is traditionally implemented using RS232, RS422, or RS485 over a variety of media.

Requirements like the Modbus routine will be called every 125mS, created major design problems a first, but were soon ironed out.

Design produce: -

- Block diagram.
- C programming using the block diagram as reference.

The first version of the Modbus communication protocol was designed to run on a PC, this was to allow for easy testing, between laptop and PC using RS232.

Functions include: -

```
void interrupt com_handler();
void trans(int c);
int readchr (unsigned char *c);
int wait_chr(char *c);
unsigned short CRC16(unsigned char *message,int length);
void modbus();
```

Week 4: Monday 14/08/2000 to Friday 18/08/2000

Connected Laptop and PC using RS232, COM1, running my Modbus test program on the PC (test.exe), and the PC version of the Modbus communication protocol on the laptop (mbpc.exe). Starting running through a detailed test producer: -

- Function 1 – Read Coil Status
- Function 2 – Read Input Status
- Function 4 – Read Input Registers
- Function 5 – Force Single Coil
- Function 6 – Preset Single Register
- Unknown function

- Bad CRC
- Short message
- Modbus reliability

Mirror bugs where found, and fixed, testing continued until I was happy that everything was working properly. Bugs where even present on the test program, which where successfully fixed. Every time changes where made to the Modbus protocol, it was imported to quickly check that the changes did not affect other parts of the program.

After everything was working properly, Modbus reliability was tested, again using my test program. This was achieved by running my random noise routines, there are several different types and all where tested and left running for at least 30 minutes. Modbus was then checked to see if it was still operational after the simulated random noise tests. (Program must never crash).

At this stage as was satisfied, that the Modbus communication protocol was ready to be converted for use on the access 4000 control panel, and testing can begin.

Week 5: Monday 21/08/2000 to Friday 25/08/2000

Converted the PC version of the Modbus communication protocol for Access 4000 control panel. Source code complied using Cosmic C, program uploaded to access 4000 panel. Two Access 4000 panel, power supply, RS232 to RS485 converter, given to me inorder to carried out functional tests.

Changes made to the technical specification; my program was modified to suit, e.g. using relative addressing instead of Absolute. The program was designed, so that changes can easily be made, and this proved to be the case as there was no problems when making the necessary changes.

The two Access 4000 control panel, with Modbus installed where connect to my PC, via the RS232 to RS485 converter. My test program was used to test the protocol. Later once a view basic tests where carried out, the Access 4000 control panels where connected up to a hardware emulator, which emulated a Genset running, AC voltage, speed, and a number of Alarms (e.g. Emergency Stop).

A series of detailed tests where carried out, the Modbus communication program was operating as expected. Changes where made to the test program display the length of time it took a slave device to respond.

Week 6: Monday 28/08/2000 to Friday 01/09/2000

Tests continued: testing in great detail; every, function, control, status, alarm, monitoring, and configuration. Making detailed notes, as report is required at a later date.

A couple more technical specification changes, meant that changes where made, program recompiled & uploaded into Access 4000 panel. Which included an additional function "function 3 – Read Holding Registers", which originally was not a support access 4000 Modbus function.

Random noise tests carried out as before using my test program, there are 6 different types of random noise included in my program, all 6 where used and left running for at least an hour in each mode. This tested Modbus reliability, making sure that the Modbus communication protocol will not affect the normal operation of the Access 4000 control panel under any condition.

Week 7: Monday 04/09/2000 to Friday 08/09/2000

The Modbus program was uploaded to an Access 4000 control panel connected to a Real Genset, tests where carried out, including basic tests like remotely starting the generator, stop the generator, emergency stop etc…

Problem: Access 4000 control panel slows down, when running continuous random noise, when the engine is running. Problem does not occur when the engine is not running. When engine is stopped, the panel returns to normal running speed.

Reason: Modbus is called every 125mS, when random noise is transmitted continuously, Modbus is activated for 80mS leaving just 55mS for normal program operation. When engine is not running this 55mS of processing power is enough to keep the panel operating normally. But when the engine is running there are lots of complex calculations being carried out, and 55mS of processing power is not long enough to finish them all, therefore the panel slow down.

Solution: There is a 1.5ms delay before characters are read, the program was redesigned inorder to make this a max timeout, which only occurs if there is no character to read. So if there is already a character in the buffer there will be no time delay, while before the program waited 1.5ms before receiving every character. This solution was implemented and was proven to work.

Testing complete, protocol working as expected.

A technical report was typed up: -

- Contents
- Introduction
- Block Diagram
- Modbus Protocol
- Testing
- Test Results – PC to Laptop
- Test Results – PC to Access 4000 using emulator.
- Test Results – PC to Access 4000 using Genset.
- Appendix 1 – modbus.c
- Appendix 2 – test.c
- Appendix 3 – mbpc.c

Week 8: Monday 11/09/2000 to Friday 15/09/2000

3 Modular control units, which include “Autostart Module”, “Basic Module” & “AC Module”, where subcontracted out and designed by FSL Electronics LTD. Before FG Wilson accepts these modules, they must be fully checked and tested, to make sure then meet are requirements.

I was given the Technical Specification for modular control units, ‘C’ Source code for all three modules, and circuit diagrams. My job was to find possible problems in the design of these modules (Hardware & Software) and to familiarise myself with the technical specification as I will be involved in the functional tests in the following weeks.

This included: -

- Reading and understanding the Technical Specification.
- Checking Source Code for possible problems.
- Checking Hardware components by download datasheet form the Internet. Check that these components will equal standards set by the “Technical Specification”.

Week 9: Monday 18/09/2000 to Friday 22/09/2000

Carried out AC Module Functional Tests: -

Description
The AC Metering module provides a voltage, frequency and current metering capability, (Series star 50Hz, Parallel star 60Hz, Single phase 50Hz, Single Phase 60Hz. The difference would only be in the covering label).

Summery Test Procedure
- LEDs Display Pattern (Volt: V1 – V8, IA: A1 – A8, IB: B1 – B8, IC: C1 – C8, Freq: F1 – F8).
- Equipment and wiring diagrams. .
- Voltage test (191V to 280V).
- Frequency test (50Hz: 45 to 55, 60Hz: 54 to 67.5).
- Three Phase Current Test, 4-bit dip switches for FSD (full-scale deflection), 16 sets of current test results.
- Single Phase current Test.
- Genset Test: connected AC module to a Genset. Set DIP switches for the Genset.
- Validation Test (Supply Voltage Test), Simulated the effect of voltage variation on the unit, confirmed the correct operation of the Unit over the specified operating voltage range of 150 to 280 VAC. Ramped the supply voltage over a 20 minutes period from 150 VAC to 0 VAC. The unit need not operate but must fail-safe.

Note: Current test, carried out using three current transformers, connected to 3 variable voltage transformers, to simulate current. Current was adjusted by changing the applied voltage to the current transformers, current was check using calibrated Tektronix THS720P Scope Meter.

Voltage test, carried out using variable a voltage transformer, the voltage was check using calibrated fuke 87 multimeter.

Frequency test carried out using calibrated function generator.

Test report was typed up, and signed by both my testing partner and I. We where given a week for these tests, finished one day early.

Week 10: Monday 25/09/2000 to Friday 29/09/2000

Monday OFF – took my first floating Holiday

Carried out Autostart Module Functional Tests: -

Description
The Autostart module provides remote start capability, 3-attempt crank, together with enchanted engine monitoring and protection.

Summery Test Procedure
- Start / Stop Tests
- Power On tests – press ROR button once, and record all LEDs and current consumption.
- Speed learning – Set over speed default values to 3500Hz. When Genset is running, record the LEDs under following two conditions: Running time <10s (FTT), Running Time > 10s (FPT).
- Crank Sequence Test – disconnect fuel supply, press start button. During the process, check the LED status and record the result.
- Faulty MPU test – disconnect MPU signal, press start button, check LEDs and crank.

- Cold Weather Start/Stop – The glow plug and the glow LED will energise for 5 sec before the engine starts to crank. Check if the sequence is OK.
- Remote Start/Stop
- Battery Voltmeter, use power supply and function generator to simulate Genset running.
- Oil Pressure, Simulate Genset running. Use a pot to adjust the sender resistance.
- Water Temperature, Simulate Genset running. Use a pot to adjust sender resistance.
- Emergency stop during engine stop.
- Emergency stop during engine running.
- Emergency stop during cranking (disconnect fuel supply).
- Low Oil Pressure.
- High Engine Temperature.
- Over Speed
- Supply Voltage Test – operating voltage range (6 – 17V).
- Voltage ramp down form 6V to 0V over a 20-minute period, unit need not operate but must fail-safe.
- Reverse Voltage Test – connected reverse voltage 17 VDC to the module for 1 hour. Afterwards check if unit still works.

Test report was typed up, and signed by both my testing partner and I, Three days was given for these tests, finished one day early.

Modified my Modbus Protocol Test Program for Access 4000 control panel (Test.c), fixed a couple of minor bugs, and made it more user friendly. The program was not originally designed for use by anyone else other than me. It was email to Hong Kong to help them test the Modbus protocol I designed for the Access 4000 control panel.

Carried out Basic Module Functional Tests, these tests followed a similar pattern to the tests carried out on the Auto start Module.

Week 11: Monday 02/10/2000 to Friday 06/09/2000

CI panel test: -

- Software development in 'C'.
- The CI panel, has a RS232 to RS485/RS422 converter, which needs to be tested, this is done by linking RX+ with TX-, RX- with TX+, which is known as a loop back test. The RS232 side of the communications board is connected to a PC via COM1.
- My program "CI_test.exe", which was designed specifically for this task, is then executed on the PC, once the test is started, the producer is completely automatic, and takes 4 seconds (if all tests pass) or 9 seconds (if all tests fail) to complete the automated tests.
- Tests include: -
  - Different baud rates, 9600 bps, 14400 bps, 19200 bps, 38400 bps, 57600 bps, and 115200 bps. Program starts at the slowest (9600 bps) and works its way to the fastest (115200 bps).
  - Three different tests are carried out on each baud rate: -
    - ✓ Test 1: Data 0 to 255 to 0 TXed & RXed with 16-BIT CRC.
    - ✓ Test 2: Random Data form 0 to 255 TXed & RXed with 16-BIT CRC.
    - ✓ Test 3: Random Data 0x00 or 0xFF TXed & RXed with 16-BIT CRC.
- The program is designed to be easy to use, as it will be used to test PCBs straight of the production line. Anyone without any specific knowledge can operate the program with ease.
- Program displays a simple pass or fail, beside tested baud rates, the user also as the option after testing is complete for a detailed report, which included details of all three test carried out on each baud rate.

- Program designed, coded and tested, by me. Works well and will be put to practical use in the near future.
- Function Include: -

```
void main_menu(void);
void Com_Test(void);
int  test1(void);
int  test2(void);
int  test3(void);
void report(void);
void interrupt com_handler(void);
void trans(int c);
int  readchr (char *c);
unsigned short CRC16(unsigned char *message,unsigned short
data_length);
void baud_rate(int rate);
void help(void);
```

Week 12: Monday 09/10/2000 to Friday 13/10/2000

Next project is a database using "Microsoft visual C++ 6.0" (ODBC Data Source, MS Access 97 Database, DAO). I have not used Microsoft visual C++ 6.0 before nor had any experience with object oriented programming.

The week was spent playing around with visual c++, creating small applications. Example source was downloaded form the Internet, including step by step worked examples. Also the MSDN Library was used as a source of reference, including sample code, documentation, technical articles, the Microsoft Developer Knowledge Base.

Step-by-step example programs included: -

1. Displaying text.
2. Adding event handlers.
3. Creating a dialog box.
4. Displaying two views.
5. Using pointers.
6. Creating a list.
7. Moving within the list.
8. Editing the list.
9. Saving the list.
10. Adding a list control.
11. Editing using hints.
12. Select the current item.
13. Using LPARAM values.
14. Adding a pop-up menu.
15. Using callback.
16. Sorting the columns.
17. Owner draw.

Step-by-step example database included: -

1. Creating a simple database.
2. C++ / Access interface.
3. Adding names.
4. Updating names.
5. Deleting names.
6. Merging the database.

Final year object oriented programming notes was download, and use as reference: -
URL: http://tree.engj.ulst.ac.uk/oop/oop.pdf
By: Dr. Colin R Turner
Date: April 18,2000

Also detailed information was found at http://www.freeskills.com/

Week 13: Monday 16/10/2000 to Friday 20/10/2000

Continued preparing for my next project, created a simple database using Microsoft Access, called "Names.mdb". This was added to the ODBC Data Source Administrator, as a System DSN using Microsoft Access Driver (*.mdb).

Using the MFC AppWizard (exe) including database support a new c++ project was created.

The program developed as follows: -

- Added form view dialog box (Child), including four edit boxes for the four fields: ID, Title, First Name, and Second Name.
- Added move controls to menu bar, tool bar and form view dialog box, adding the appropriate functions for the controls.

- Added controls “Add”, “Edit”, “Delete” and “Delete All”, along with code to make them function, and popup dialog boxes for “Add” & “Edit”.
- Other controls included: “go to”, “Search”, “Sort”.
- List control added, along with a new class “CDataBaseListView”, this class displays the complete database in the form of a table. Controls added to allow the list control window to be load.
- Windows message handler added, for column click, right mouse click, left mouse click etc… code added to allow the database to be sort when a column is clicked. Also a popup menu appears when you right click on an item in the list, allows you to Del/Edit the select item. Also if an item in the list is clicked it become the current item in both the list & form views.

Screen dump of the test database program shown below: -

The test database program, which I designed & coded form scratch, give me a little practice using Microsoft Visual C++ 6.0. This test database program may be small, but it give me a lot of experience and knowledge, as this program made use of many basic features included in visual C++, including dialog boxes, edit boxes, menu bar, tool Bar, list ctrl, buttons, combo, Database (CRecordset & CDaoRecordset) AfxMessageBox and many more.

At the end of the week I felt I had gain a great deal of knowledge about programming with Microsoft Visual C++ and was know capable of creating a number of small applications, while just two weeks earlier I had no knowledge (had never used C++ before).

Week 14: Monday 23/10/2000 to Friday 27/10/2000

Was given a copy of the existing database “Panel.mdb” created in Microsoft Access 95, since my program must be backward compatible (e.g. must load the old databases), I started writing the code to access the database through a visual C++ program, using the DAO database interface. The reason why I used the DAO interface and not the ODBC Data Source Administrator is because DAO lets you get direct access to the *.mdb file, and does not require the data source to be added to the ODBC before hand, hens it is possible to select the *.mdb file just as you would any other if the default path cannot be found or the user selects open in the file menu. The existing database was large and complex, so it was going to take some time to display all the appropriate views/tables.

Most of the data is stored in a table called [tblSETTINGS], this table contains 135 columns and at present 230 records, in order make life easy of the operator this table was split into 8 views (Commercial, System, Calibrate, Setpoints, Inputs, Outputs, Security, Main) a Multidocument child window was required for each view.

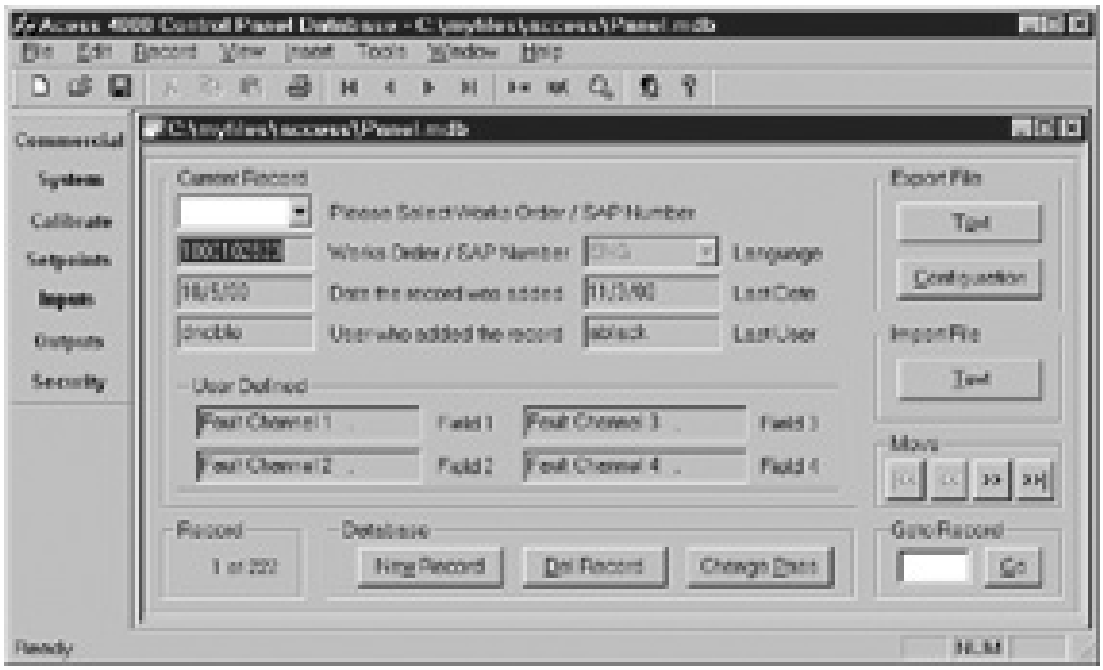
Using Combo boxes, Edit boxes, Check boxes, Radio Buttons, and many others the all eight form view was designed and working, included a save function which saved changes to the database. Also dialogs for adding, deleting, records where also added.

Security was high in importance; the database is to have a 3-level (DE, TE, Admin), username and password security system, the existing database already had several tables, which where for user control and monitoring. Table [tblusersstatus] contained the fields "Username", "Level", "Status", "Lastlogged", and "Password Changed Date". This table had to be auto updated in the background without the users knowledge. The table [tblUSERHIST] contains the fields "LastDate", "LastUser", "Issue", "WON", this table stores the user history, e.g. every record the user views/modifies is store here, at present there are 2500 records. These two tables where added to my program and automatic code to use/update these tables was added.

Also by this time you needed a password & username to login, all usernames are stored in the table [tblusersstatus]; the password was encoded onto the database using various encoded techniques. Another option was to stored the password in the windows registry, but this had the draw back that, that if the user when to a different machine there system would no longer have a record of his/hers password, and this was ruled out early on. Storing the password on the database meant that any user could use any computer, as the password was not stored locally, put actually encoded onto the database.

Because the system uses a 3 level security system, some users have restrictions imposed on them, e.g. only users with level-3 Admin Access or the user who created the record in the first place can delete it. Also starting working on a Class that viewed the User History/User List table (including deleting/adding users), only level-3 Admin operators will have access.

Screen dump of the main section show below: -



Week 15: Monday 30/10/2000 to Friday 03/11/2000

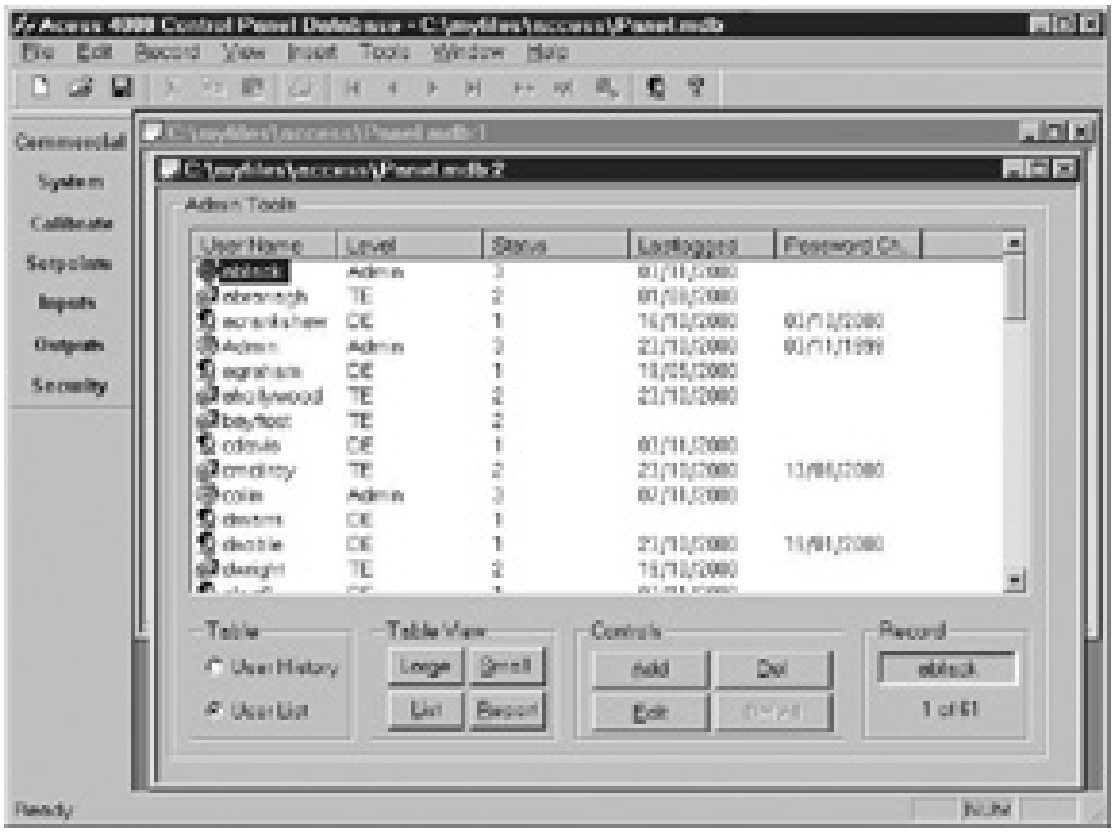
Added admin tools to database: -

- Display a list of all users.
- Display User History.
- Add a user.
- Edit a user.

- Del a user.
- Clear User history.

Only users with level 3 (Admin) Access will have access to the admin tools.

Screen shot of the admin section shown below, this is how it appears at present, will properly be modify in the coming weeks, so the appearance may change: -



On Tuesday of this week, I attended the training course “Control Systems” in the CEC. The course length was 7½ hours, and went into considerable detail on most of FG Wilson’s control panels. Including Circuit diagrams, ISO standards, Price, Quality, functional description, advantages, disadvantages, defects, components etc… control systems, looked at in detail included: -

- Engine Interface Module.
- Starter Motor Solenoid (EIM SR)
- Glow Plug (pre-heat)
- Fuel Control Solenoid
- Safety relays
- 1001 Control System, (drawing D19443B)
- 2001 Control System, (drawing D20449A)
- LCP0, LCP1. LCP2 Control Panels (drawings D18494F, D20025B, D22640B)
- 4001 & 4001E Control Systems (drawing D18495G, D19516D)
- Ribbon/Field Cable Interface (drawing D22628C)
- Remote Annuciators
- Automatic Transfer Switches
- TC – Transfer Switch
- Mti – Transfer Switch
- Access 4000 Control Panel
- 6000 Series Control System

SCI Annuciator Data Frame: -