

**24-7 Loader Mainboard Rewrite**

**Software Design Document**

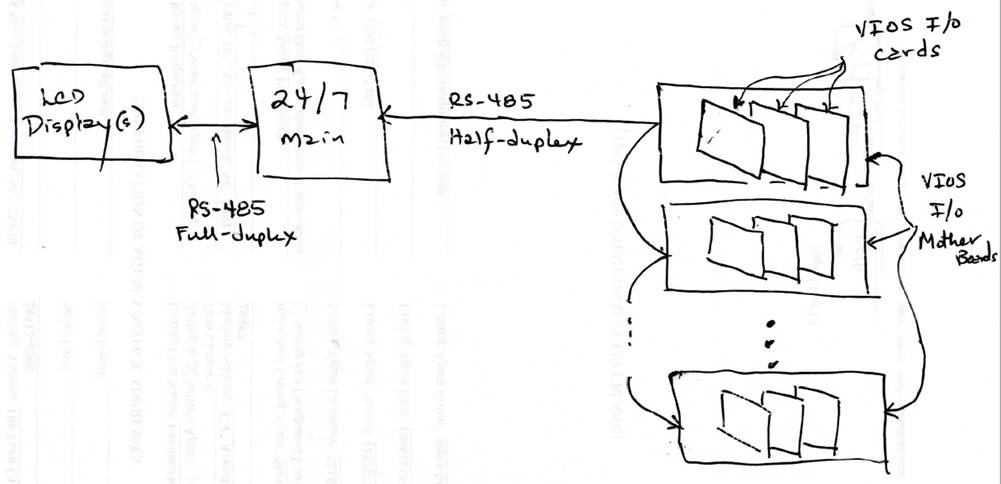
|  |  |
| --- | --- |
| **Name** | **Email** |
| Chen Liu | chen.liu@unadyn.com |

1. Introduction
   1. Motivation

The mainboard is getting old and they have part availability issues eventually. The 24/7 is targeted towards a different customer than FACS, and less money consuming than FACS.

* 1. Overview

The following image show the basic layout of the whole system:



The project only needs to rewrite the 24/7 mainboard program, the mainboard will also be update on electronics level. The two communication parts (RS-485 Full-deplex and RS-485 Half-deplex) should remain the same.

1. Design Considerations
   1. Old version issues
2. Limited RAM on the original chip (LPC2387), some C code (less RAM utilization).
3. USB port that share with RS-232 are not useful.
4. No FACS interface communication on old version.
   1. New feature update
5. Processor update: LPC2387 to LPC1768.
6. Remove FDI programming header J7, all uC programming will now be done via the RS-232 port.
7. Remove all old USB stuff that was never implemented.
8. Change U2 128k\*8 12C EEPROM to 32k\*8 FRAM (faster).
9. Make the new USB interface on SCI0 identical to that on the FOCUS-PROX.
10. Make the new CPM5000 interface on SCI1 identical to that on the FOCUS-PROX.
    1. Environment minimum requirement

Everything is written in C++.

Add “USES\_LPC1768” conditional compile feature.

* 1. Design methodology

1. Architecture

There are 3 parts need to be programmed for the 24/7 whole system:

1. Mainboard, one system should have one mainboard which working as the “Brain” for the system, this project mainly points to mainboard rewrite and keep others same remain the same.
2. IOMotherboard, even older microcontroller embedded system.
3. Display, two chips need to be programmed. One is Amulet display chip, another is ARM LPC2387 chip.
   1. System design
      1. Only .h
      2. PCB
      3. Loader
      4. IoCard
      5. Station
      6. Pump & Pump swap
      7. Valve
      8. AppAlarmSystem
      9. Display
      10. Discrete
   2. Communication design
      1. SCI0 (ARM programming via RS-232 or Host USB Communication)
      2. SCI1 (CPM-5000 FACS Module)
      3. SCI2 (RS-485 Full Duplex to touchscreen LCD)
4. **System**: softwareVersion, startupstatus, onboardClock, TopviewLayout, serviceDiagnostics, DebugPointerValue
5. **Loader**: KnownConfig, DisoveredConfig, FinalConfig, LaoderRuntime, HardwareError, Passwords, AlarmSetClear, AlarmsCleared, RuntimeStationAssignments, SilenceAlarmOcurred
6. **Device**: installation, name, ioStates
7. **Pump**: OptionalHardware, Enable, StationAssignments, StationAssignmentChange, Settings, Vacuum, OpsStatus, CurrentStationStatus, QueueStatus, Alarms, Alarms, Runtime, Statistics
8. **Station**:
9. **PumpSwap**:
   * 1. SCI3 (RS-485 half Duplex to VISO PCBs)
10. From Main to All VIOS

Io mother DiagCmd

1. From single VIOS to MAIN

Board type and function

Io mother status

Io card status

* + 1. I2C0 (2-wire interface to FRAM chip memory)

1. Data Design
   1. Non-volatile RAM design (EEPROM -> FRAM)
   2. RAM design (use as less as possible)
   3. Message design (remain same)

There are three fixed protocols:

1. **DisplayCommsProtocol**: (including header + body + footer)

\*\* #0 - <DLE>

\*\* #1 - <SOH>

\*\* #2 - Message type

\*\* #3 - Message Number

\*\* #4 - <DLE>

\*\* #5 - <STX>

\*\* === MESSAGE BODY ===

\*\* #? - <DLE>

\*\* #? - <ETX>

\*\* #? – Checksum

1. IoBoardCommsProtocol: (mainly use for IoBoard to Mainboard communication)
2. LoaderAccessProtocl:
   * 1. Between display and mainboard
     2. Between IoMotherboard and mainboard

The motherboards have unique addressing, which is how serial communication messages between the main and individual motherboards are routed.

Mainboard to motherboard commands: outputs

Response from motherboards define the type of I/O cards installed: input (only contains the map and its inputs and outputs states)

1. Component Design
   1. CPU code
      1. CPU Pinout

Aasa

//------------------------------------------------------------------------------

//-- PORT0

//-- b0(46) - N/C

//-- b1(47) - N/C

//-- PCB196: b2(98) - TxD0 - RS-232 serial comms, SCI #0

//-- PCB196: b3(99) - RxD0 - RS-232 serial comms, SCI #0

//-- PCB196: b4(81) - SCI #3: EN-MOTHER-Tx~Rx

//-- b5(80) - N/C

//-- b6(79) - N/C

//-- b7(78) - N/C

//-- b8(77) - N/C

//-- b9(76) - N/C

//-- PCB196: b10(48) - DSPLY-TXD2 RS485 Full Duplex, SCI #2

//-- PCB196: b11(49) - DSPLY-RXD2 RS485 Full Duplex, SCI #2

//-- b12 - N/C

//-- b13 - N/C

//-- b14 - N/C

//-- b15(62) - N/C

//-- b16(63) - N/C

//-- b17(61) - N/C

//-- b18(60) - N/C

//-- b19(59) - N/C

//-- b20(58) - N/C

//-- b21(57) - N/C

//-- b22(56) - N/C

//-- b23(9) - N/C

//-- PCB196: b24(8) - AD0[1] input - VBAT2ADC

//-- PCB196: b25(7) - ~ALRM-YEL(GP)

//-- PCB196: b26(6) - ~ALRM-RED(GP)

//-- PCB196: b27(25) - I2C0 - SDA to serial EEPROM

//-- PCB196: b28(24) - I2C0 - SCLK to serial EEPROM

//-- PCB196: b29(29) - RTS0- USB - serial comms SCI #1 to USB chip (this is a g/p

output!)

//-- PCB196: b30(30) - Output (GP) - unused, but must be an output in order for

b29 to function correctly

//------------------------------------------------------------------------------

//------------------------------------------------------------------------------

//-- PORT1

//-- PCB196: b0(95) - Input (GP) - ~INVAL-232 line

//-- PCB196: b1(94) - Input - USB Flag

//-- b2 - N/C

//-- b3 - N/C

//-- PCB196: b4(93) - Output (GP) - 7-segment display, decimal point

//-- b5 - N/C

//-- b6 - N/C

//-- b7 - N/C

//-- PCB196: b8(92) - Output (GP) - 7-segment display, 'C' segment

//-- PCB196: b9(91) - Output (GP) - 7-segment display, 'D' segment

//-- PCB196: b10(10) - Output (GP) - 7-segment display, 'E' segment

//-- b11 - N/C

//-- b12 - N/C

//-- b13 - N/C

//-- PCB196: b14(89) - Output (GP) - 7-segment display, 'G' segment

//-- PCB196: b15(88) - Output (GP) - 7-segment display, 'F' segment

//-- PCB196: b16(87) - Output (GP) - 7-segment display, 'A' segment

//-- PCB196: b17(86) - Output (GP) - 7-segment display, 'B' point

//-- PCB196: b18(32) - CTS0- USB - serial comms SCI #0 to USB chip (this is a g/p

input!)

//-- b19(33) - N/C

//-- b20(34) - N/C

//-- b21(35) - N/C

//-- b22(36) - N/C

//-- b23(37) - N/C

//-- b24(38) - N/C

//-- b25(39) - N/C

//-- b26(40) - N/C

//-- b27(43) - N/C

//-- b28(44) - N/C

//-- b29(45) - N/C

//-- PCB196: b30(21) - blown fuse, g/p input

//------------------------------------------------------------------------------

//------------------------------------------------------------------------------

//-- PORT2

//-- PCB196: b0(75) - SCI #1 TXD1 line - serial comms to CPM5000 (Neuron)

//-- PCB196: b1(74) - SCI #1 RXD1 line - serial comms to CPM5000

//-- PCB196: b2(73) - SCI #1 CTS line - serial comms to CPM5000

//-- b3(70) - N/C

//-- PCB196: b4(69) - ~CPM-Reset, from CPM5000, set to interrupt on a high-to-low

transition, g/p input

//-- PCB196: b5(68) - ~Reset-CPM, reset signal to CPM5000, g/p output

//-- PCB196: b6(67) - ~HRDY to CPM5000, g/p output

//-- PCB196: b7(66) - SCI #1 RTS - IO4

//-- PCB196: b8(65) - TxD2 - SCI #2 serial comms to USB chip

//-- PCB196: b9(64) - RxD2 - SCI #2 serial comms to USB chip

//-- PCB196: b10(53) - ~Boot/Run control line

//-- PCB196: b11(52) - ~Option line, g/p input

//-- b12(51) - N/C

//-- b13(50) - N/C

//------------------------------------------------------------------------------

//------------------------------------------------------------------------------

//-- PORT3

//-- b25(27) - N/C

//-- b26(26) - N/C

//------------------------------------------------------------------------------

//------------------------------------------------------------------------------

//-- PORT4

//-- PCB196: b28(82) - SCI #3: MOTHER-TXD3

//-- PCB196: b29(85) - RXD3 - SCI #3: MOTHER-RXD3

//------------------------------------------------------------------------------

* 1. Main PCB board
  2. VIOS Motherboard
  3. IO card
  4. Display

1. Diagram
   1. Class diagram
   2. Sequence diagram
   3. Deployment diagram
2. Appendices
   1. Refactor tips
      1. For EEPROM:

Using class members instead all passing whole thing.

* + 1. Getter and Setter:

For the getters, add const keywords to make sure not change the object.

Eg: int GetValue () const {}

Getter and setter normally one line of code, no need to use {} in next line

* + 1. Inheritance

Main purpose is to reduce the code duplication

* + 1. Visibility

All members are private, using setter and getter to manipulate value, but make sure to follow the correct inheritance.

* + 1. Reduce memory usage

Try to make non-static, dynamic allocate?!

Use pointer instead of real value, especially to passing pointer or address to functions, methods…

Not passing whole project, only passing address or pointer:

Void PrintEntity(const Entity& e){} // must use &reference instead of real instance

* + 1. Minor enhancement

#ifndefine xxx

#include “xxx”

#endif

Change to #praga once, for header file

* 1. CPU usage
  2. Task priority

**WatchdogTimerUpdate**: Update the CPU's watchdog timer.

**ProcessIncomingIoMotherData**: Process individual bytes of data, then completed incoming messages.

**IoCommsUpdate**: Send the next outgoing message to the I/O motherboards.

**ProcessIncomingDisplayData**: Process individual bytes of data, then completed incoming messages.

**StationInputsTimeUpdate**

**PumpInputsTimeUpdate**: Read/debounce the inputs.

**LoaderLogicTimeUpdate**: Time update for loader operations.

**StdUiTimeUpdate**: Time update task for the UI.

**NonvolMemTimeUpdate**: Compares the contents of shadow RAM and stores updated data in the nonvol memory chip.

**PcbErrorCodeTimeUpdate**: Update the displayed error code.

**UpdateIoMotherLossOfCommsWatchdog**: Update the timer that protects against loss of comms with the I/O motherboards.

**UpdateDisplayLossOfCommsWatchdog**: Update the timer that protects against loss of comms with the display board.

**LdrAccessIntfTimeUpdate**

**AlarmMgrSilenceAlarmTimeUpdate**: Implements the timing required for the Silence Alarm function.

**ValidateClockTimeUpdate**: Validate the onboard clock.

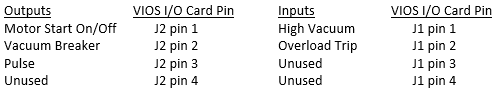
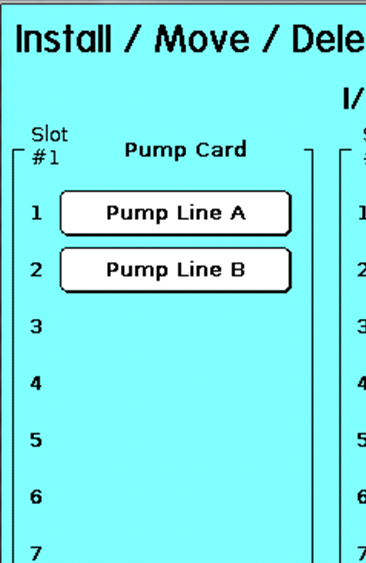
**StartupTimeUpdate**: Tasks that checks for the end of the hardware discovery sequence and keeps the loader access interface updated with that status.

**SendAllAccessIntfTimeUpdate**: Task that handles the work of sending data in response to a 'SendAll' request.

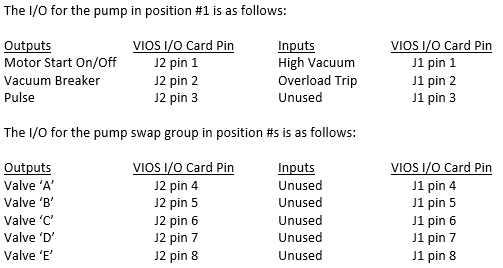
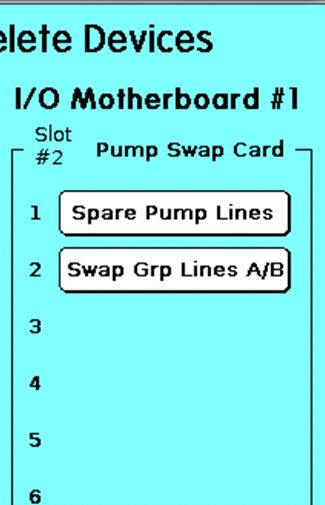
**BufferPoolDiag**

* 1. VIOS I/O card

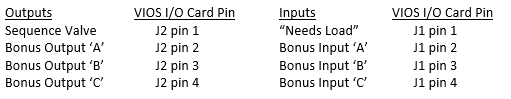
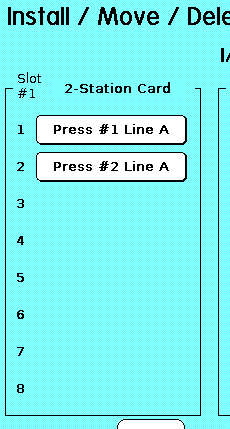
1. **VIOS Pump Card**



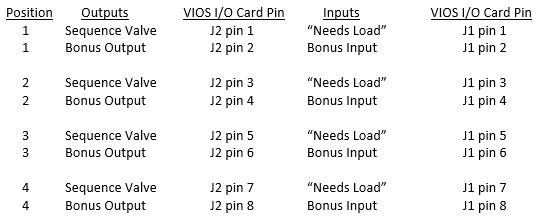
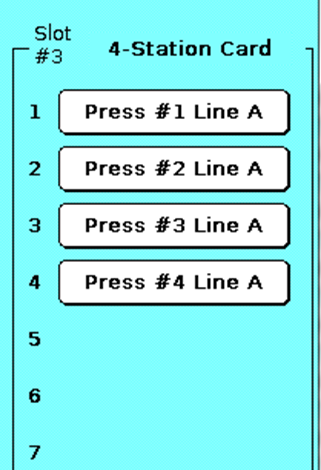
1. **VIOS Pump Swap Card (not usual)**



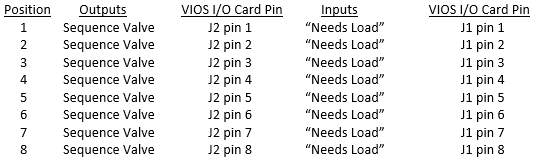
1. **VIOS 2-Station Card**



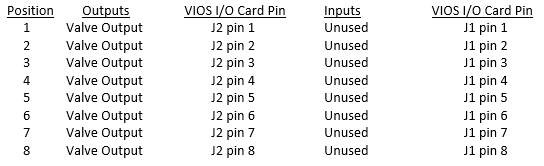
1. **VIOS 4-Station Card**



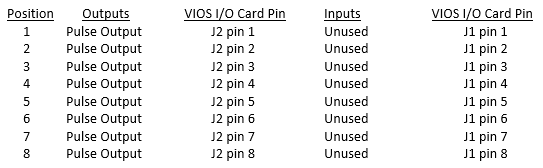
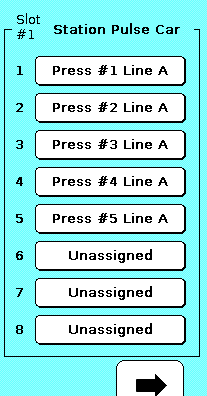
1. **VIOS 8-Station Card**



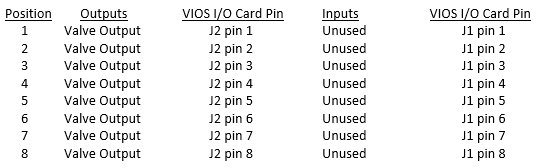
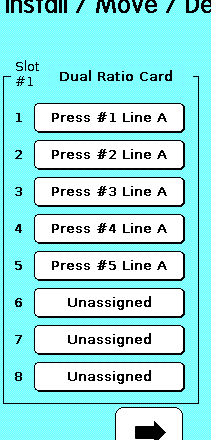
1. **VIOS Global Purge Valve Card**



1. **VIOS Station Pulse Card**



1. VIOS Station Dual Ratio Card



* 1. RoadMap

|  |  |
| --- | --- |
| 02242020 |  |
| Bytebuffer |  |
| Scheduler |  |
| Task | Pcb196SchedulerTaskId |
| FloatRound |  |
| PoolBuffer = old PoolByteBuffer |  |
|  |  |
| 02252020 |  |
| CallbackRegistry / list |  |
| Formatable |  |
| PcbVoltage / ADC STUFFS | AnalogInputRaw / R16Running |
| FRAM = old EEPROM |  |
| Pcb7SegDisplay |  |
| CPM5000 |  |
| VNC(USB) |  |
|  |  |
| 02262020 |  |
| UiPasswords = old UiPasswords |  |
| Alarm / Actuator / installable |  |
| Switching stuffs/ sensors / output/input |  |
| Level sensor |  |
| Alarmoutput = actuator |  |
| AlarmManager |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |