C/ORNL99-0546

CRADA Final Report
for
CRADA Number ORNL99-0546

Dual Manifold System for Arraying Biomolecules

M. J. Doktycz
Oak Ridge National Laboratory

J. Johnson
Innovadyne Technologies, Inc.
(formerly Rheodyne)

Date Published:

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
Managed by
UT-Battelle, LLC
For the
U.S. Department of Energy
Under contract DE-AC05-00OR22725

APPROVED FOR PUBLIC RELEASE

UNLIMITED DISTRIBUTION

CRADA Final Report
for
CRADA Number ORNL99-0546

Dual Manifold System for Arraying Biomolecules

M. J. Doktycz
Oak Ridge National Laboratory

J. Johnson
Innovadyne Technologies, Inc.
(formerly Rheodyne)

Date Published:

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
Managed by
UT-Battelle, LLC
For the
U.S. Department of Energy
Under contract DE-AC05-00OR22725

APPROVED FOR PUBLIC RELEASE

UNLIMITED DISTRIBUTION

Dual Manifold System for Arraying Biomolecules

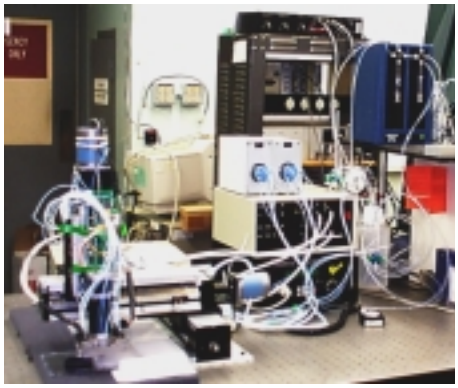
(CRADA No. ORNL99-0546)

Abstract

The objective of this CRADA is to establish a new approach to fluid transfer and array construction. This new approach will involve a high-speed, multiplexed fluid distribution valve and ink jet valves. It will enable the parallel handling of multiple reagents for a system that will have multiple applications in addition to the high-speed construction of microarrays. The primary tasks involve proof of principle experiments aimed at establishing key components of the technology and evaluating various optional configurations. The basic platform for evaluating the technology will be set-up by the Contractor at Oak Ridge National Laboratory (ORNL) and will employ custom valving prepared by Rheodyne. The test platform will consist of a motion controller, 3-axes of motion, software, and pneumatic control; and will be used to evaluate the hybrid valve.

Overview

The goal of this cooperative research and development agreement was the construction of a test stand for evaluating a hybrid valve. The valve consists of a fluid switching mechanism with integrated ink jet dispensing components. The CRADA partner constructed this custom valve. The test stand was developed at ORNL and was designed for performing simple procedures, such as aspirating and dispensing reagents between microtiter plates, using this custom valve. The test stand consisted of several pieces of hardware including a Velmex VP9000 motion controller, a Lintech 3-axis motion control system, two Rheodyne LabPro fluid selectors, a Hamilton ML900 syringe pump and a custom ink jet driver module (power supplies and driver circuit) used for driving the Lee ink jet valves. A helium tank and pressure regulator provided the system pressure. Control of the instrument is via a personal computer containing a digital input/output board and four addition serial ports. The primary deliverables of the CRADA project were the test stand (pictured below) and the software for controlling the instrument. The software is described in detail below.



Software Description
The software for running the Omega Test Stand is written in MS Visual Basic 6.0 with Service Pack 3.0 installed. The software is comprised of Form and Basic modules written by Steve Hicks at ORNL. This code makes use of 2 DLL's for communicating with specific ComputerBoards and Rheodyne hardware. The software 'as shipped' from ORNL does not include these DLL's which are assumed to be already installed on the PC test stand computer. The DLL from ComputerBoards is used to communicate with and control the DI/O card, which in turn is used to pulse the Ink-Jet valves via the Valve Driver Box. The DLL from Rheodyne is needed to command the Rheodyne Omega and selector valves.

VB Software:

Micro_Arrayer.vbp Project File

Project Name **(File Name)**
Description

Form Code:
frmManualRheodyne **(**ManualRheodyne.frm**)**
Form that allows testing of the Omega and Selector valves (manual control)
frmR_InkJet_ValveDrv **(**R_InkJet_ValveDrv.frm**)**
Form that allows testing of the Ink-Jet valves (manual control)
frmRobo_HW_Setup **(**Robo_HW_Setup.frm**)**
Form used to configure hardware (Match hardware layout with software input)
frmRobo_Main **(**Robo_Main.frm**)**
Main Run Screen for Hardware Setup, Program Editing, or Program Executing
frmRobo_ProgEdit **(**Robo_ProgEdit.frm**)**
Form that allows selection of saved programs and editing and testing (Dry Run) of programs
FrmRobo_ProgRUN **(**Robo_ProgRUN.frm**)**
Form that allows the execution (Starting, pausing, stopping) of programs
frmRobo_Splash **(**Robo_Splash.frm**)**
Boot-up Form
frmSerialC **(**SerialC.frm**)**
Form that contains the MSComm1 and MSComm2 (Microsoft Comm control 6.0)
Comm1: Velmex VP9000 Comm2: Hamilton ML900
frmDeckLayout **(**DeckLayout.frm**)** <**Not currently used/finished**>
frmDispenseMap **(**DispenseMap.frm**)** <**Not currently used**>
frmOptions_ProgData **(**frmOptions_ProgData.frm**)** <**Not currently used/finished**>

Basic Modules:
CB_Driver **(**16&32cbw.bas**)**
This file contains the Visual BASIC declarations for ComputerBoards Hardware products.

DelayTimer **(**DelayTimer.bas**)**
Provides a source of delay for controlling program flow (timing)

'Public Sub Delay_Timer(PauseTime As Single)
'Public Sub WaitFor(PauseTime As Single)

DIO_CompBoards **(**DIO_CompBoards.bas**)**
Functions for communicating with/controlling the DI/O board
'Public Function Config_DIOBoard() As Integer
'Public Sub WritePort(PortNum&, DataValue%)

Disk_Access **(**Disk_Access.bas**)**
Functions for opening/closing and writing to a debug file
'Public Sub File_Open()
'Public Sub LogData(STR As String)
'Sub File_Close()

GeminiGV6 **(**GeminiGV6.bas**)**
'Public Function GeminiConnect(iPortNum As Integer)
'Public Sub GeminiDisconnect()
'Public Sub GeminiInitialize()
'Private Sub SendCommandGV6(sCmd As String)
'Public Sub GeminiReadResponse(Response As String, Optional ExpectedResponse As Variant)
'Public Sub StopMotionGV6()
'Public Sub Home_XYZ()
'Public Sub Move_X(ByVal xPos As Long, ByVal spd As String, ByVal acc As String)
'Public Sub Move_Y(ByVal yPos As Long, ByVal spd As String, ByVal acc As String)
'Public Sub Move_Z(ByVal zPos As Long, ByVal spd As String, ByVal acc As String)
'Public Sub MoveDone(iAxis As Integer)
'Private Function GV6Velocity(iAxis As Integer) As Single
'Private Sub GeminiReadVelocity(Velocity() As Single)
'Public Function Position(iAxis As Integer) As String
'Public Sub GeminiReadAllPositions(Position() As Long)
'Not Used! Public Function GeminiGV6Status(iAxis As Integer) As String

HML900_Serial **(**H_MicroLab900.bas**)**
Functions that communicate/control the Hamilton Syringe pump
'Public Sub ConfigSerialPort_HML900()
'Public Sub AutoAddress_Hamilton()
'Public Sub Initialize_HamiltonML900()
'Public Sub SendCommand(CmdStr As String)
'Public Function Read_ML900() As String
'Public Sub CloseSerialPort_HML900()

InkJet_DIO **(**InkJet_DIO.bas**)**
High-level functions that control the firing of the Ink-Jet valves via (DIO_CompBoards.bas functions) the
DI/O board
'Public Sub Pulse_Valve(SpikeTime As Integer, HoldTime As Integer, ValveNum As Integer)
'Public Sub Hold_Valve(SpikeTime As Integer, HoldTime As Single, ValveNum As Integer)
'Public Sub Hold_Valves(SpikeTime As Integer, HoldTime As Single, G1_CHSel As Integer, G2_CHSel As Integer)
'Public Sub Pulse_Valves(SpikeTime As Integer, HoldTime As Integer, Group As Integer)

Motion **(**Motion.bas**)**
'Public Sub StopMotors()
'Public Sub HomeXYZ()
'Public Sub MoveX(ByVal Position As Long)
'Public Sub MoveY(ByVal Position As Long)
'Public Sub MoveZ(ByVal Position As Long)
'Public Function Read_Position(ByVal Axis As Integer) As Long
'Private Sub MoveXYZ(X_cnts As Long, Y_cnts As Long, Z_cnts As Long)

Program_Run **(**Program_Run.bas**)**
Main Module with functions that Executes the ROBOTIC Micro_Arrayer Programs
'Public
' Sub Read_Filenames()
' Sub Read_HardwareSetup_File()
' Sub Load_Program_to_Memory()
' Sub Program_Execute(ByVal Mode As Integer)
'
'
'Private
' Sub Parse(strCmdTyp As String, Data1 As String, Data2 As String, _
Data3 As String, Data4 As String, Data5 As String)
' Sub Execute_ST(Mode As Integer)
' Sub Execute_PF(Mode As Integer)
' Sub Execute_WF(Mode As Integer)
' Sub Execute_WA(Mode As Integer)
' Sub Display_Step(Message As String)
' Sub Aspirate(Vol As Single)
' Sub Syringe_Side_Wash(TDur As Single)
' Sub Report_Plate_Pos(Plate_Pos As String, X As Integer, Y As Integer, Z As Integer)
' Sub Report_Wash_Pos(X As Integer, Y As Integer, Z As Integer)
' Sub Report_1stWell_Offset(Well_Pos As String, X As Single, Y As Single)
' Sub ParseStr(InString)
' Sub Check_Pause()

' Function CalculateX(X_Deck As Integer, Off_X As Single, Index As Integer, Operation As String) As Long
' Function CalculateY(Y_Deck As Integer, Off_Y As Single, Index As Integer, Operation As String) As Long
' Function CalculateZ(Z_Deck As Integer, Off_Z As Single, Movement As String) As Long
' Function CvrtVol_to_Time(ByVal Vol As Single) As Integer

Read_Write_Files **(**Read_Write_Files.bas**)**
Functions that read and write files to disk (note: not exhaustive list – file reading/writing takes place in
other routines as well but it is mainly of the debug nature)
'Public
' Sub WriteStartFile()
' Sub ReadHardware(Fname As String)
' Sub WriteHardware(Fname As String)
' Sub LogData(STR As String)
' Sub File_Close()

Rheodyne **(**Rheodyne.bas**)**
Module that has code communicating with Omega and Selector Valves via Rheodyne DLL function calls.
'Public
' Sub Initialize_OmegaValve()
' Sub Initialize_SelValve()
' Sub RotateOmegaValve(nPosition As String)
' Sub RotateSelValve(nPosition As String)
' Sub Close_OmegaValve()
' Sub Close_SelValve()
' NOT implemented!
' Function Read_Omega_Position()
' Function Read_Selector_Position() As String

Robo_Var_Struct **(**Robo_Var_Struct.bas**)**
Module that contains Global variables and Structures for holding hardware and program data

VP9000_Serial **(**VP9000_Serial.bas**)**
Module that contains low and high level code communicating/controlling the operation of the X-Y-Z
stages via the Velmex VP9000 Step Motor Controller
'Public
' Sub ConfigSerialPort1(PortNum As Integer)
' Sub SendCommand(CmdStr As String)
' Sub StopMotion()
' Sub Home_XYZ()

' Sub Wash_XYZ(X As String, Y As String, Z As String)
' Sub Move_X(ByVal xPos As Long, ByVal spd As String, ByVal acc As String)
' Sub Move_Y(ByVal yPos As Long, ByVal spd As String, ByVal acc As String)
' Sub Move_Z(ByVal zPos As Long, ByVal spd As String, ByVal acc As String)
' Function MoveDone() As String
' Function Position(iAxis As Integer) As String
' Sub CloseSerialPort1()
' Function VP9000Status() As String
'==========
==========

Software Operation:
Upon startup the software opens a file that must be in the current directory called StartFile.txt.
"A384_D384.hws","PF_R1.pgm" This file contains the names of two more files. The first name has the extension hws, standing for hardware setup. Following a comma, is the second file name that indicates the last program file that was being used the last time the software was run. This file has the extension pgm, standing for program.

A suggested naming convention would follow this pattern. A384_D384.hws meaning Aspirate from a 384 well plate, dispense to a 384 well plate. The data in this file is entered on the Hardware Setup Form. This file is written by the program and should not be edited. (But if one really feels the need to, it can be edited with a text editor. Just make sure that the last line does not contain any characters after the last data field. Spurious characters visible or not can cause the reading program to crash. It will try to read another line and expect to find 5 more data fields, not just the extra character.)

Hardware File:
The hardware file is rather self-explanatory after a look at the hardware setup screen. The general format of the data fields is X, Y, Z positions. All of the plate positions have two Z fields. The first Z is the Aspirate height, the second Z is the Dispense height. Important note: Z increases as the dispense head moves down!

Example A384_D384.hws
"Locations of Deck Hardware, Z heights, and axes move parameters"
"A",4916,2083,3600,2600,"S384W"
"B",4916,8865,3600,2600,"T384W"
"C",15004,2083,2600,2600,"T384W"
"D",15004,8865,2600,2600,"T384W"
"E",25014,2083,2600,2600,"T384W"
"F",25014,8865,2600,2600,"T384W"
"Wash",400,7890,2200
"R1eservoir",5504,100,3550
"R2eservoir",15730,100,1500
"Park",10000,50,1550
"ZClear",2200 **‘Clearance height (Z) of tallest obstacle on the deck**
"TBlock",6,2 **’12 tips**

"Xp","6000","50" **'X speed and acceleration**
"Yp","6000","50" **'Y speed and acceleration**
"Zp","2000","10" **'Z speed and acceleration**

Program Files:
The four example programs provided with the software follow a similar descriptive convention. Examples of operations in increasing complexity: Wash.pgm performs a Wash [WA] of all tips. WF_Sys.pgm performs a Plate fill [WF] water fill with the system fluid. PF_R1.pgm does a Plate Fill [PF] operation and aspirates from Resevoir1. ST_PlateA.pgm performs a sample transfer one to one aspirating from PlateA, dispensing to any or all other plate positions. All program files contain on the first line the type of operation that is to be preformed enclosed in brackets []. The remaining lines all contain 6 data fields separated by commas. Even if 6 data fields are not needed for a particular task such as a Wash, placeholders must be used separated by commas to fill out the line. The first field is the task field. If it is a Aspirate or Dispense task, the next field contains the Plate location, followed by the starting well location (Always A1 until this is implemented), followed by the number of blocks in the X-directions, followed by the number of blocks in the Y-direction, finally followed by the amount to aspirate or dispense. For aspiration the number corresponds to uL. For dispensing, the number corresponds to uSeconds. A Wash task only has one field with significance, the first one, which indicates how many seconds to keep the Ink-Jets open during a wash. When editing be careful about spurious characters on the last line, the remarks above apply to these files as well.

Example ST_PlateA.pgm
[ST]
Aspirate,A,A1,2,1,20
Dispense1,C,A1,2,1,1500
Dispense2,B,A1,2,1,1500
Wash,1,0,0,0,0

Top Level Menus:
<u>Hardware Setup</u>:
The hardware setup screens are used for entering plate locations on the working deck area. This screen also gives access to manual test screens that allow manual operation of the valves and syringe pumps.

<u>Program Edit</u>:
The main purpose of the Program Edit screen is to allow the operator to pick and view a program before executing the program. Choosing a dry run, which includes all the motion, but not fluid operations, will test the program operating sequence.

<u>Program Execute</u>:
The Program Execute screen is for running tested working programs.

Ink Jet Dispense System Hardware

Valves

- **Ink Jets, 1200 Hz Valves (INKX0502600AB, Lee Co)**

Valve Driver Box

- **Control circuit: Spike and Hold design**

The driving circuit is based on a spike and hold method suggested by the valve manufacturer. This circuit significantly reduces the opening time and shortens the closing time as well for the valve. A certain voltage is needed to actuate (open) the valve. Once the valve is open, a lower voltage can be used to maintain (hold) the open state. For faster opening times, this minimum voltage can be exceeded. The valve coil can withstand short duration high voltage excitation several times higher than the normal opening voltage. The high voltage excitation will quickly overheat the solenoid windings and therefore can not be used to operate the valve for extended times (multiple seconds, like during a wash cycle) because resistive heating will quickly melt the winding. Thus, the term "spike and hold" accurately describes the operation of the valve driving circuit. Because nothing is gained by the higher voltage after the valve is opened, all pulses, even the sub millisecond ones are divided into a short (~150us) high voltage excitation followed by a low voltage hold signal for the remainder of the total desired ON (open) time, or pulse width. With a high voltage of 40 volts, the valve can reliably be operated with 0.2 ms pulses. Therefore, all excitation is applied in this two-stage manner, i.e. a short 40V spike (~150 us) followed by a hold voltage of ~ 8 volts for the remainder of the open time.

Most of the logic for controlling the valve driver outputs is contained in an Altera CPLD chip. The programmable logic chip provides the link between the digital outputs from a Digital I/O card in the PC and the high and low side driver IC's that apply power and ground to the Ink Jet valves. This chip has most I/O pins assigned, but ample reserves of logic cells for additional logic changes or added features.
The circuit board has extra auxiliary drivers for other 12 VDC loads.

Power supplies

- **Two Power One linear regulated power supplies are used. The high voltage spike voltage is supplied by a 48 VDC supply. The hold supply is derived from a 1 Amp, 8-Volt 3-terminal regulator. The TTL logic is derived from a 1 Amp, 5 Volt 3-terimnal regulator. Both of these regulators are powered from a 12 VDC power supplies.**

Software Known issues:

1) **Currently the Valve Correction Factors are set to 1 and are not stored to disk like they should be. This will be implemented soon.**
2) **Program editing will be implemented on the Program Edit Form soon.**

Notes:

(1) The software contains periodic checks on whether an operator has hit a “Stop” or “Pause” button. This is not a substitute for an ‘emergency stop’ function, which all moving machines should have. The options were discussed with Velmex. The recommended method for implementing an emergency stop is to run the Limit Switch common grounds (WHT and BLK wires) through a Red Mushroom-head 2-position, maintained switch. When this switch is pressed, the controller will sense all limits are active and prevent movement in both directions on all axes immediately. This hardware method provides immediate protection that this software can not provide and continues to function even in the event of software failure.

(2) This software is capable of running 4 fluid handling operations – Sample transfers, plate fills, system (water) fills, and tip washing (one tested sample program of each type is included). It has a graphical user interface that is functional, but not extensive or polished. For example programs can be edited with a text editor, as well as on the program edit screen. This is allowed because a GUI editor is not implemented. A GUI editor could prevent the operator from making mistakes, thus making the program more robust.

C/ORNL 99-0546

INTERNAL DISTRIBUTION

1. M. J. Doktycz, 4500S, MS-6123
2. R. C. Mann, 4500S, MS-6124
3. J. C. Miller, 4500S, MS-6125
4. K. M. Wilson, 111B UNV, MS-6499
5. Laboratory Records, 4500N, MS-6285 (RC)
6. Office of Technical Information and Classification, 4500N, MS-6285

EXTERNAL DISTRIBUTION

1. P. A. Carpenter, Department of Energy, Oak Ridge Operations Office, Post Office Box 2008, Oak Ridge, Tennessee 37831-6269

2. DOE Work for Others Office, MS G209

3. Innovadyne Technologies, Inc. 2835 Duke Court, P.O. Box 7329, Santa Rosa, California 95407
    - J. Johnson, Director of Technology
    - J. McComb, President
    - J. Noonan, CEO