Addendum to Franklin J. Rudolph's Resume: Detailed Case Studies Arranged In Approximate Chronological Order, Describing His Career.

At Beacon Power Corporation:

I designed and led the implementation of the hierarchical control system for a "flywheel farm". It consisted of

- 200 high speed flywheels connected in clusters of 10 flywheels
- 20 clusters of 10 flywheels each that are controlled by a Linux-based "Master Controller"
- 1 Linux-based "Supervisor" that communicates with the electric utility grid authority (ISO) via DNP3 and MODUS, and decomposes short-term power requirement commands from the ISO into appropriate commands to each Master Controller in order to optimize the performance and availability of the "farm" as an energy storage and recycling facility.

This system was designed as a distributed Linux-based software package consisting of several hundred lines of C/C++ and Linus (bash) scripting. It is a muti-process, multi-threaded, distributed application that relies on heavily interconnected TCP/IP socket streams communicating over several protocols (Ethernet, MODBUS, DNP3, CANBUS), and also involves several GUIs implemented with LabView and Visual Basic.

I architected the communications protocol between the Master Controller and the flywheels through the "Beacon Interface Board" (BIB), which has a TI TMS320F812 DSP on it. I designed the real-time round robin scheduler, A/D conversion schemes and fault handling on the DSP.

I architected the communications protocol that acted as the switchboard between the "frontend" CANBUS on the DSP, which connects to the Master Controller and the "backend" CANBUS on the DSP which communicates with the flywheel inverter (ECM).

I designed the interrupt handler for the DSP board which used a counter capture module to count edges from light-reflective tape segments on the spinning shaft of the flywheel, in order to calculate the speed in RPM of the shaft, I had to design an algorithm that adjusted the required resolution of the capture unit on the DSP dynamically, so it could have high resolution at low speeds and high speeds as well. The result was considered for a patent, but did not make the cut because its monetary value was not deemed significant enough to pursue.

I also designed a new median filter for which I wrote a patent disclosure, which is used in the DSP to throw out "outlyers", and can extract a near perfect signal from a forest of uncorrelated noise outlyers (which are prevalent in shaft rotation counter interrupt algorithms) and which doesn't have to explicitly track the age of signals in order to effectively identify outlyers.

I also had to design an algorithm that allowed our 10 flywheel nodes to continuously calculate their time slices on the CANBUS in order to solve a congestion problem that was making the CANBUS completely ineffective with a large number of nodes and a high error frame rate (due to the extreme electromagnetic noise in the inverter/trailer environment).

Part of the solution of the problem of data corruption in such a noisy environment involves the use of electrical to photo-optic transduction, which the BIBs and Master Controllers use to transmit CANBUS over long distances in the control trailer.

The BIB is also a data concentrator which brings together analog and digital signals from sensors directly on the flywheel and allows the GUIs attached to the Master Controllers to display real-time process information and fault conditions to operators.

ALL of these developments required extensive use of oscilloscopes, spectrum analyzers, logic state analyzers and CANBUS protocol analyzers to get the whole system to work.

I also collaborated in the hardware design of the BIB and worked extensively in the integration of hardware and software.

I worked from beginning to completion of the design and at times worked alone, but managed a team of up to 5 other software and firmware engineers during the project completion.

We used MATLAB for modeling, LabView and VB for GUIs, Wonderware for our data historian, Spectrum Digital emulators and TI Code Composer for real-time coding and debugging, gcc and CentOS 5.3 Open Source Linux for high level programming and Accurev for source revision control, all of which I am fully competent at using. We use Acrobat, Word, Excel, Visio for documentation and SQL and Access for database needs.

We also use a variety of RTUs for ISO communications and I have become conversant with ISO methods of communications with generators.

I have co-published cutting edge research papers in the field of energy storage on the electric utility grid.

The flywheel farm is now operating in Stephentown, NY, and was constructed overa a period of 18 months at a cost of $69 million. It currently provides New York State with about 10% of its frequency regulation support. The rest is mostly provided by the Niagara Hydro-electric Plant.

At Schneider Electric:

I worked as the de-facto lead of the firmware team for the Twido, a new compact, very inexpensive Programmable Logic Controller (PLC). This product used a Mitsubishi M16C CPU. The operating system was a hand crafted non-preemptive, round-robin scheduler, for which I designed and implemented the kernel. I also designed an on-device database server that acted as the interface through which all HMI functions could view the internal configuration and behavior of the PLC. I wrote the technical specs for and implemented the code for the clock functions, the schedule block function (a complex user accessible scheduler that allowed non-technical users to easily program calendar based PLC behavior using a simple HMI), and performed most of the system integration and optimization. I designed the assembly/C interface protocol and implemented simple interfaces that exploited the M16C architecture allowing calls from ASM to C and C to ASM without adversely affecting real-time performance. I had to work closely with the communications protocol designer to be certain that the interrupt drivers of the MODBUS protocol engine did not adversely affect real-time performance. I coordinated the implementation and integration of two major functional areas of the design that were implemented by another consulting firm operating remotely from France who shipped their finished code to me for integration. I worked closely with management to help set up and maintain the ClearCase source control system and used ClearQuest extensively to track bugs during the acceptance testing phase. This project was finished on schedule and under budget.

At Monarch Instrument:

I worked here as the Software Manager. I managed a team of 5 developers. As an individual contributor, I worked on a redesign of their DataChart paperless strip chart recorder. Their product draws a stripchart digitally on an endless strip of electronic paper. The executive ran on a Motorola 68331 CPU. I had to modify the existing (most assembly language) firmware to accommodate an entirely new updated set of lower cost peripherals. I redesigned the mass storage drivers to use ATAPI Zip Drives and ATA semiconductor "hard-disks". I redesigned the RS232 and SPI clock chip interfaces, the video driver and added the IDE interface, TCP/IP stack, and a proprietary packet based secondary communications processor protocol (SCP) for that allowed the 68331 to communicate with the Rabbit chip, which housed the TCP/IP and MAC/PHY protocol engines. In this position I was deeply involved in the development of low-level protocol stack TCP/IP implementation on the Rabbit. At higher level in the project, I worked on the embedded webpage application that presented a virtual instrument on the internet using the HTTP protocol.

At Gilford High School and Souhegan High Schools:

During a period of employment as an AP Math and Physics Teacher at Gilford and Souhegan High Schools, I kept my engineering skills fresh through involvement all four years in USFIRST Robotics Teams. In this capacity I spent 6 weeks each year on a team of engineers and students designing and implementing a complex remotely operated robot and several weeks additionally planning for the project. I was actively involved in every aspect of the design from running gear to real-time control system software. I mentored student designers, teaching them how to program closed loop servo control software for the steering mechanisms of the robots using the STAMP BASIC controller included in the FIRST parts kit. I also taught them how to design variable speed drive motor scheduling to reduce motor wear and tear during starting and stopping and how to schedule steering control to mimic the behavior of a differential steering mechanism using variable steering sensitivity as a function of steering azimuth. During this period I also taught scientific computer programming in C at UNH Manchester.

At MKS Instruments:

I was in charge of developing AI and neural networks based adaptive pressure controllers for semiconductor processing. The software development involved porting a large body of undocumented C/C++ and assembly code from an Intel 80196 platform to an 80188EB platform as well as supporting existing code in the field. The work involved a custom designed real-time multi-tasking preemptive operating system and a complete rewrite of the application layer. Mastery of MATLAB, MathCAD (for model simulation), CorelFLOW (for documentation), Borland C++, Paradigm Debug and Windows were required in this effort. In addition Scientific Word and MS Word as well as Lotus 1-2-3, MS Works, MS Excel and numerous other software tools were extensively used.

I also served on the MKS Technology Transfer Team, the Concurrent Product Development Program's Definition and Development Proposal Specifications Team and

chaired the Software Process Improvement Team which was then moving the company toward CMM level 3. MKS is ISO-9001 certified.

At UNH:

As a Computer Science Masters Degree student and later an Electrical and Computer Engineering Doctoral Student I studied artificial intelligence, parallel processing and artificial neural networks in robotics. I worked in the UNH Robotics Lab, (DARPA/ONR grant N00014-89-J-3100) under Dr. W. Thomas Miller. I also investigated the fault tolerance of neural network implementations (Analog Devices Career Development Grant) under Dr. Michael Carter. My dissertation topic investigated a method of path planning and control for redundant mechanisms that used direct inverse modeling. Prior to my work, it had been strongly asserted that direct inverse modeling was not a powerful enough paradigm to do machine learning for redundant mechanisms. My dissertation explored the boundary conditions of this assertion and provided concrete methods of structuring a learning architecture that combined collections of locally optimizing artificial neural nets called CMACs (a particular type of sparse distributed memory) and heuristic constraints that allowed direct inverse modeling to work quite well at solving this problem. The motivation here was that direct inverse modeling is more efficient than other competing paradigms and CMACs are computationally more efficient and deterministic than exponentially complex methods like multi-layer perceptrons.

Tools, methods and fields of study: Robot simulation, MATLAB, MathCAD, Kalman Filters and other estimators, closed loop feedback control systems, developing discrete controllers from analog designs using Z-transform and bi-linear transform, Artificial Intelligence, Lisp, Compiler Design, direct closed loop servo control, motion control, rate feedback versus derivative feedback, linearizing non-linear systems, neurophysiology, Artificial Neural Networks, high dimensional state space analysis.

Summary of activities managed over 13 year period as owner/president of Cybertronix Corp.

Throughout much of my career I have been fully responsible for all aspects of the management of a small, closely held engineering consulting corporation, including procuring and specifying contracts, and managing personnel. During this period, I also designed and implemented software for a wide variety of applications, involving principally hardware controllers and interfaces. These included type font digitizers/undigitizers, various disk controllers, process controllers, databases, text editors, compilers and an industrial robot controller and path planner. I used most processors and operating systems commonly available during that period including FORTH, Pascal, C, Lisp, Modula-2, FORTRAN and BASIC.

At Sanders Associates:

At Sanders Associates I spent several months designing and implementing an automated language translator in support of their 458B Electronics Countermeasures test bench. My

project implemented a method for converting a large body of assembly language software for their proprietary MIPS-16 processor to 8086 assembly language. I did this by constructing a set of TECO macros that acted as a lexical analyzer to recognize and replace MIPS-16 instructions with ASM86 macros that mimicked MIPS-16 instructions. TECO was a DEC macro text editor. I used scripts written in the TECO command language to modify large libraries of code in batch runs without human intervention. The interesting part of this project was that the MIPS-16 and the 8086 had radically different processor architectures so the TECO macros were quite complex. The project saved Sanders time and money by avoiding a lot of human error in the tedious manual translation of code. I was able to do the project entirely with on-hand, low-tech software tools. The debugging and acceptance test was less than a month long.

At Butler Automatic:

I designed a system of autonomous robots that moved rolls of paper around the floor of a large high-speed printing (multiple) press operation. The system consisted of robotic hoists and robotic vehicles which communicated with each other via an Arcnet coaxial highspeed network (between the hoists) and infrared LED/photosensors pairs between the hoists and vehicles to send commands and pass management data. The IR based RS232 interface was implemented entirely using a high priority task tightly coupled to the operating system's tick counter. It time-sliced generation of 300 baud square-wave pulses of IR light and interrupt generated recognition of the sensed pulses to send packets of data between hoist and vehicle. What was difficult here was the low band-width of the processor (a 4 MHz Z80). The real-time operating system had to be tuned to an extremely high degree of efficiency for this to work in parallel with the control algorithms and continuous diagnostic self-tests. Another serious problem was that our compile-time started consuming more than an hour for each build. By using the same high-speed network over which the robots communicated to host a virtual hard-drive for the compiler, our compile time was reduced to minutes and the project was saved. The original development system had only floppy disks so I simply rewrote the CPM86 based disk interface to communicate over the network with an IBM-PC to which the development system became a parasite. The project was completed without an emulator, using variable dumps and a lot of detective work to find bugs. I designed the controls and communications for the system of hoists and my business partner at the time, Dr. Tamas Hetenyi, designed the autonomous guided vehicles (AGVs).

The high speed network described above was implemented over an ArcNet network (a deterministic time-sliced network protocol that was an early competitor of Ethernet). I developed all the code that interacted between the control system and the Arcnet. The entire application was designed with strict compliance with the OSI 7 layer standard.

At USM Corporation (Dyna/Pert Division)

I worked on the system 9000, which was a central data controller for a distributed network of Dyna/Pert robotic insertion machines. My job involved surveying the then emerging market of pre-PC single board computers and available operating systems to

select a processor (Intel 8086), single board computer (iSBC86/12) and operating system (polyFORTH). Next, together with two other designers, I designed a complete multi-user/multitasking system that would allow up to 32 users to log onto a single 8086 based computer and simultaineously edit, download and execute insertion programs to any of the insertion machines on the network. The network was a proprietary RS-232 protocol. My job was to design the editors for each of the different types of insertion machines, design and implement the floppy disk and hard disk drivers for the system and modify the operating system for our particular collection of hardware and real-time interrupt requirements. This project also involved PDP-8 programming.

At Compugraphic Corporation:

I programmed, in 8080 assembly language, a font disk manufacturing station that would produce font diskettes for their AdVantage line of phototypesetting systems. I also worked on an RSX11M operating system that managed their database of high resolution digitized type fonts. My task there was to reverse engineer their library of high resolution type fonts and develop a vectorization and redigitization algorithm that would allow them to convert their entire font library from a high resolution form to a lower resolution library to support a new lower cost line of photo-typesetters that used a lower resolution. This project, written in PDP-11 assembly language and RSX11 scripts allowed Compugraphic to reduce the cost of implementing the new product line by avoiding the need to rescan all the font images at a lower resolution (a very time consuming and expensive manual task).

At Hamilton Test Systems, Windsor Locks, CT:

I developed a diagnostics program aimed at assisting maintenance personnel in isolating problems in elevator drive control systems and car and hall call logic sequencing.

At McDonnell/Douglass Corporation:

I developed software for a very large scale three dimensional engineering design drafting system, call CADD, operating on an IBM 370 mainframe. During this time I worked on two major design projects. The first was to design, in FORTRAN, a program that took as input an arbitrary solid shape, bounded by a collection of parametric bi-cubic patches, and a set of sectioning planes that cut the solid. The cross-sectional section drawings that were generated by the cuts had to be arranged on an engineering drawing sent to a flatbed plotter. The program had to follow good drafting practice in arranging the sections and labels that identified each cut bi-cubic surface were arranged around the sections such that good engineering drawing practice was followed and no leader lines crossed.

The next major project I worked on was a design module called FLUIDS that allowed a designer to lay out an arbitrary 3D network of metal tubing in an arbitrary bounded 3D space in such a manner that the network of tubes were manufacturable. All metallurgical properties and physical dimensions of the tubes had to be automatically looked up and design rules followed to assure that the tubes would not collapse, kink or be fatigued by

the manufacturing process required to route them. This was implemented in assembly language and FORTRAN.

I also worked on several problems related to the numerical accuracy of surface intersections.

As a flight test engineer on the Tomahawk Cruise Missile program I debugged real-time assembly language programs written for a Litton 4516C CPU running the inertial navigation and final targeting systems for the missile. I worked from assembly listings of the program during on-range assignments at White Sands Missile Range and Dugway Proving Grounds on a missile mockup aircraft (a Beechcraft King Air) that used the Tomahawk navigation system as the flight commander the pilot followed during mock missile missions. I had to determine defects on-site, collaborate with designers in St. Louis, design hand patches for the program and install them. I performed these same duties as test engineer in the St. Louis lab facilities during static simulations between missile range tests. I also had to troubleshoot hardware and software problems during range tests to keep the missions functioning.