A NEW HIGH-EFFICIENCY STEPPER MOTOR DRIVER FOR OLD TECHNOLOGY STEPPER MOTORS*

N. Greenough, C.C. Kung
Princeton Plasma Physics Laboratory, Princeton, NJ 08543-0451
ngreenou@pppl.gov.

***Abstract*— Stepper motors have been used to drive mechanical systems for many decades. Stepper motors require a current-limited multiphase driver, and techniques have evolved from simple resistive drivers to complex high efficiency, high-performance switching devices. The first high-efficiency switching designs were patented over 30 years ago and have become widely used throughout industry. This has caused a shift in driver technology from a quad-drive system to an H-bridge system, and a change in motor design from four windings to two.**

There are a great many older systems with the earlier motor technology still in use. Retrofitting such systems can take one of several paths:

- **Replacing the entire mechanical drive motor system, its driver, and reworking all the associated communications protocol between the driver and the outside world;**
- **Choosing a high-efficiency "H"-bridge driver, re-working its communications protocol and raising the drive voltages to hazardous levels to get the older-technology motor to perform adequately;**
- **Designing a high-efficiency, high-performance motor driver system that will work with older-technology motors and existing communications protocols.**

This paper presents a high-efficiency, high-performance stepper motor driver and that provides all the advantages of a modern motor driver, yet works with existing older-technology stepper motors. It also solves a well-known failure mode of the standard "H" bridge drive system, and will pay for itself in decreased electrical energy costs compared to an old-technology resistive driver. An application-specific communications controller and motor driver will be described to create a drop-in replacement for an existing controller-driver system.

I. INTRODUCTION (*Heading 1*)

The modern approach to high-efficiency stepper motor drive systems is a switch mode H-bridge driver configuration feeding a two-winding, four-wire motor. However, there are a great many systems in the field still equipped with low-efficiency resistive drivers and four winding (center-tapped) motors. Converting the older-style driver requires the use of excessively high voltage H-bridge to overcome the winding inductance of the motor. The solution to this dilemma is a different configuration of high-efficiency controller that can operate with an older-style center-tapped high-inductance stepper motor. This paper describes the design of such a controller and its intelligent interface to an external control system. It was commissioned by General Atomics to replace failing stepper-motor drive systems on radio-frequency transmission-line tuners for the high power Fast Wave equipment on DIIID.

II. BACKGROUND

The modern approach to high-efficiency stepper motor drive systems is a switch mode H-bridge driver configuration feeding a two-winding, four-wire motor. However, there are a great many systems in the field still equipped with low-efficiency resistive drivers and center-tapped two winding, 6-wire motors. Converting the older-style driver requires the use of excessively high voltage H-bridge to overcome the winding inductance of the motor. The solution to this dilemma is a different configuration of high-efficiency controller that can operate with an older-style center-tapped high-inductance stepper motor. This paper describes the design of such a controller and its intelligent interface to an external control system. It was commissioned by General Atomics to replace failing stepper-motor drive systems on radio-frequency transmission-line tuners for the high power Fast Wave equipment on DIIID.

The conversion of an older style center-tapped motor system to a modern high-efficiency controller has several challenges:

- The older motor winding inductance is high, requiring hazardous drive voltages if driven by an H-bridge controller.
- There are few high-efficiency switching controllers available for center-tapped motors.
- There is considerable expense and effort in replacing both motor and controller.
- The computer programs or hardware that calculate the inputs to the motor driver system may need to be redesigned or re-written for a different communications protocol.

It is possible to modify the design of an H-bridge switching controller once the current paths of the motor and controller are understood. The modern H-bridge controller uses one of several current-control techniques to switch and maintain winding current at design value. When a motor step is desired, the bridge controller turns on opposite pairs of transistors or switch devices to place the entire power supply across the motor winding. The winding current builds rapidly

*This work supported by the US DOE Contract No. DE-AC02-09CH11466

due to the relatively high voltage and the motor inductance. A current sensor then detects when the winding current reaches design value. When detected, the drive voltage is switched off and an idling circuit is placed around the winding. The idling circuit recirculates the winding current which decays slowly due to winding losses. When the winding current has decayed a small value, the controller recharges the current by again switching the same opposite pair of transistors on for a brief period of time. Control system designs differ in the aspects of current sensing, switching and timing. One of the simpler and more successful designs is implemented in the National Semiconductor LMD18200 series of controller ICs [1]. The datasheet also contains an excellent description of its operation, and can be used as background for the following discussion.

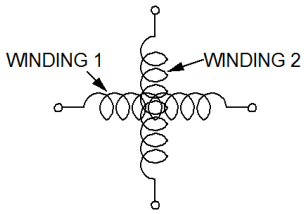

Fig. 1. Modern 4-wire motor

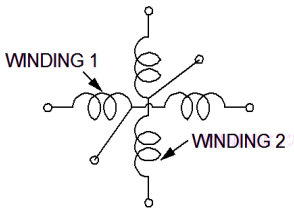

Fig. 2. Older 6-wire motor

Some points to consider:

- A 4-wire motor winding is driven end-to-end and reversed for stepping while a center-tapped 6-wire motor is unidirectionally driven by half-winding.
- The 6-wire motor's undriven half-winding must not be loaded or the motor torque will be reduced.
- The 6-wire motor's undriven half-winding end will rise to twice the power supply voltage. It is a coupled transformer.
- Current recirculation is achieved in a 4-wire motor controller by turning on upper or lower pairs. Recirculation can also be achieved for a center-tapped motor by appropriate isolating diodes and switching devices.
- It is common to double-drive the motor windings for smoother torque. Both windings are active simultaneously.

The following diagrams show one of the 2 windings in both types of motors. Current flow is shown in red. There are two phases to a high-efficiency driver: current drive and recirculate.

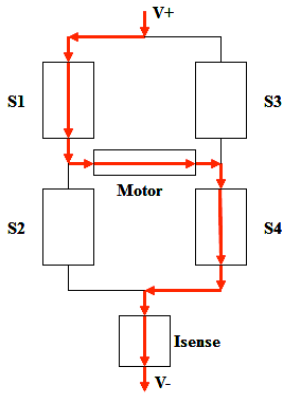

Fig. 3. Modern 4-wire motor, current drive mode. S1 and S4 ON, S2 and S3 OFF.

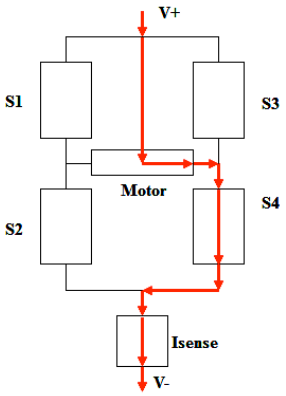

Fig. 4. Older 6-wire motor, current drive mode. S4 ON, S1, S2 and S3 OFF.

The controller transitions to recirculate mode as shown in Figs. 5 and 6 when the winding current reaches design value. This is accomplished by turning OFF the switches that connect to the V+, V- power supply and turning ON switches that create a current loop around the motor. The inductance of the motor sustains the current.

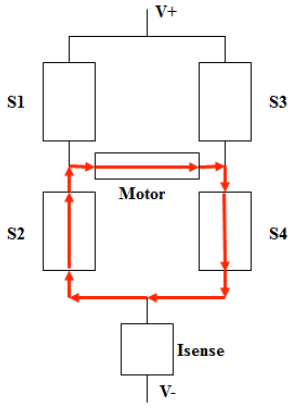

Fig. 5. Modern 4-wire motor, recirculate mode. S2 and S4 ON, S1 and S3 OFF.

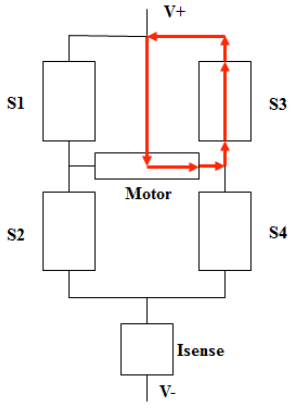

Fig. 6. Older 6-wire motor, recirculate mode. S3 ON, S1, S2 and S4 OFF.

To take a step, the motor winding current is reversed on one of the windings for the 4-wire configuration, and swapped to the other half of the center-tapped winding for the 6-wire configuration. The switches are driven in a mirror image of Figures 3, 4, 5 and 6.

Figure 7 shows a simplified circuit that implements the switching and current paths of Figures 4 and 6. Note that it appears similar to a conventional H-bridge except that the current path through the upper switch S1 is reversed. This is accomplished with an isolating Shottky diode for low loss. Note that there is no longer a direct active semiconductor path across the V+ and V- power supply. This increases reliability as the possibility of destructive current flow is reduced should the drivers misbehave.

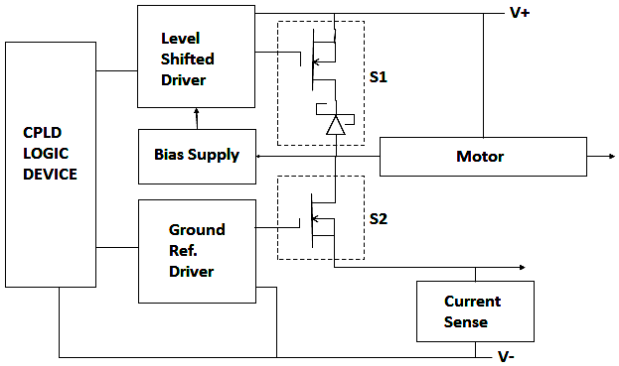

Fig. 7. Simplified ½-bridge circuit, for 6-wire motor

The upper FET needs a bias voltage raised above the V+ power supply for proper operation. The undriven motor winding rises to 2x the V+ power supply as a consequence of the switching current regulator. Small Shottky rectifiers and a simple Zener regulator supply the needed bias voltage for the upper FETs from the motor winding ends.

III. Current Sensing and Control Logic

The current sensor located in the return to the motor power supply detects that driven motor current has reached design value. This event triggers the FET switch drive logic to change over from current drive mode to recirculate mode. The logic implements a fixed-off-time algorithm that estimates the decay of motor winding current. The logic transitions back to the current drive state when the off-time expires. The off-time is typically set to 300- 400 microseconds.

An XC95108 CPLD logic device [2] and quad comparator implement the switching logic and current sense. The FET gate drivers are implemented with discrete components. A complementary NPN-PNP pair provide sufficient drive current to overcome the FET gate capacitance in about 50-100 nanoseconds. All of the above circuits are implemented on a printed-circuit board and packaged in a single 12 x 18cm enclosure. The enclosure serves as the heat sink for all the power devices. A fan is not required as the total power dissipation is just a few watts in operation.

IV. Intelligent Controller

The second enclosure houses the intelligent half of the system. Its function is to accept commands from an external computer system and interpret them into a simple timed step and direction stream for the motor controller's CPLD. An existing specification for the equipment being replaced provided a starting point for the intelligent interface.

The intelligent interface interprets commands received from a remote computer system over an RS485 interface, and knowing the present position of the stepper motor, calculates a number of steps and direction for the stepper motor to achieve the desired position. In this case, the intelligent controller section was designed to emulate the command set of an existing controller. Commands such as "go to position absolute", "go to position relative", "recalibrate", "request present position" and "request error status" were implemented within the General Atomics RF antenna stub-stretcher tuner system. The original intelligent controller was implemented with an 8051 microcontroller and a PC board-full of peripheral ICs. A modern PIC microcontroller was selected that contains all of the required peripherals and more on-chip. The PIC16F887 is a 40-pin chip that provides enough I/O pins to directly drive a small ASCII liquid-crystal display, five user control buttons and all the analog and digital I/O required to support the CPLD motor controller described above. The PIC16F887 contains flash memory for the control program, static RAM for program variables and non-volatile memory for storage of setup constants.

The intelligent controller has three main interfaces:

- An operator interface for direct control by personnel for setup, troubleshooting and manual operation if desired
- A computer interface for receiving calculated stub and stretcher motor positions, either RS485 or Ethernet
- An optically-coupled step-and-direction interface with the motor driver

The operator interface is provided by a simple menu-and-button selection system. The menus are displayed on a 2-line by 16-character LCD and the operator makes selections and specifies quantities with five buttons. Three of the buttons are positioned beneath the LCD display and are labeled dynamically by the running control program within the PIC processor. Two buttons are positioned to the right of the display and are primarily used by the operator to make numeric choices.

The menu-based operator interface allows the operator to set up the controller for a specific stub or stretcher unit without recompiling the internal control program. Specifics such as the RS485 address or maximum allowable motor position are set in this manner. The setup specifics are stored in non-volatile memory within the microcontroller and are automatically restored at power-up. The operator can also send the stepper motor to its inner and outer limits for electromechanical setup, cause a recalibrate command to synchronize the controller with the mechanical end-stops, step the motor a small amount in either direction or send the motor to a specified position.

The computer interface section of the intelligent controller program was written to follow an existing bit-frame-based specification. Each command has a set number of bytes that varies with the command, followed by a check byte. An "on-the-fly" state machine interpreter parses the incoming command stream into its command, data and check bytes. The command is executed upon correct reception of the check byte.

The motor controller interface consists of four optocouplers, one each for a step pulse, direction control, run current set and accelerate current set. The step pulse is a timed stream of short pulses, one for each requested step of

the motor. The direction bit is high for CW rotation, low for CCW rotation. The default motor current is holding current and is set at about ½ of rated winding current. This is sufficient to restrain the stepper motor from being mechanically back-driven by the weight of the drive mechanism. Motor dissipation is extremely low in this state. The motor current is raised to the rated running current when the "Run" signal is activated. This is the manufacturer's rated operating current for smoothest operation of the motor and rated temperature rise. The motor current is raised somewhat over ratings briefly when the "Accelerate" signal is activated. This is intended to overcome the sticky-friction of the drive mechanism when the motor has been idle for an extended period of time.

Acceleration is programmed into the intelligent controller to obtain the fastest available travel time given the mass of the motor rotor and drive mechanism without skipping steps at startup. The step rate starts out slow and advances to a more rapid rate. Deceleration is also employed as the motor nears its final position. This allows the motor to be stopped quickly without over-running its destination position.

V. On-Chip Program

The on-chip program that implements the user interface, computer interface and motor driver interface is written in a 2-thread configuration for a PIC16F887 processor [3]. One thread is activated upon a fixed-rate interrupt generated by an on-chip timer. It is used for time-critical tasks such as motor step timing, interfacing with the on-chip serial port, driving the LCD display and reacting to mechanical over-travel. The other thread occupies the remaining execution time and handles such as computing steps and direction, interpreting and reacting to commands received from the computer interface and running the menu-based operator interface.

The on-chip program is written directly in the assembler language of the PIC processor [4]. Programming directly in the Microchip assembler allows complete control of the chip and the path of the executing code and is the best choice for time-critical applications.

There is little benefit from including the on-chip firmware listing in this paper as it runs to about 100 pages of assembler code. There are a few interesting techniques in the code that may be of interest to readers when code size goes above the 2 kilo-word page boundary of the basic PIC architecture.

A. Handling Interrupts and the 2 kilo-word Page Boundary

In practicality, the address space of the PIC program memory is partitioned in 2 kilo-word blocks due to the program addressing limit of 11 bits in jump or subroutine call instructions. A special register is supplied (PCLATH) to allow the programmer to place and access code beyond the base 2 kilo-word block. Interrupts, however, complicate the issue as the PCLATH register contains the high bits of the interrupted code address, and not the location of the interrupt service routine.

One solution is to place a stub interrupt handler in the base memory block starting at the interrupt vector location (usually 0004). Save W, PCLATH, FSR, STATUS and anything else into known data memory locations. Remember that the current data memory page is also unknown when saving W, so use a common data memory area that appears in all data memory blocks. All PIC processors that have paged data memory have a common area for this purpose. Then set PCLATH to the proper memory block where the interrupt routine code resides and jump or call that location. At the end of the interrupt routine, set PCLATH back to the base memory page and jump or return to the interrupt stub in the base page. Restore all saved registers including PCLATH and execute the RETFIE (Return from Interrupt) instruction to recover the correct pre-interrupt executing code location from anywhere in memory [5].

B. Placing Code Beyond the 2KWord Page Boundary

Placing code in the higher regions of the PIC address space has aptly been named "jumping into hyperspace" in homage to a certain late 1970s science-fiction film. Doing this is easiest to accomplish if the main program code does not need to leave the base page, and only subroutines need to be placed in the upper areas. Point PCLATH to the correct page and "call" the relevant routine. The return is automatically handled by the PIC, but PCLATH will remain pointing to the subroutine's page. Forgetting this detail will cause the running program to jump to the incorrect page if another subroutine is called in any page including the base page. The results can be spectacular and debugging this error is difficult as not even the PIC's hardware debuggers can follow it. It is best to explicitly force PCLATH to the proper subroutine page before each subroutine call.

VI. Conclusions

A design for a high-efficiency driver for older-style stepper motors has been presented. An understanding of its switching technique may be of interest for similar retrofit designs. The original resistive motor driver dissipated more than 200 watts continuously and required a fan to cool it. In contrast, the new design switching driver dissipates only about 10 watts including its power supply and is fanless.

An intelligent motor controller based on a modern microcontroller has been presented. Design details and schematics are available from the author for those interested in further information.

VII. References

[1] The latest LMD18200 data sheet is available from www.ti.com

[2] The latest XC95108 data sheet is available from www.xilinx.com

[3] The latest data sheet for the PIC16F887 is available from www.microchip.com.

[4] The Microchip assembler and programming environment are available from www.microchip.com.

[5] See www.piclist.com/techref/microchip/pages.htm for further info.