

A hydraulic servo controller using the IX1 microprocessor

A.C. Wheatley, N.J. Nelson

Abstract

There's a good chance that the car you drive contains parts made on a machine controlled by Forth. This paper presents a servo controller implemented using the IX1 microprocessor and programmed in Forth. One of these servos controls each axis of a hydraulic robot. The robots are widely used in the automotive industry for producing stretch-formed parts.

A.C. Wheatley B.Sc., A.M.I.E.E.
N.J. Nelson B.Sc., C.Eng., M.I.E.E.
Micros Electronics Ltd.,
Units 4-5, Great Western Court,
Ross-on-Wye, Herefordshire.
HR9 7XP U.K.
Tel. +44 1989 768080
Fax. +44 1989 768163
Email. acw@micros.co.uk

Introduction to stretch-forming

Stretch forming is a process used to create parts for cars, coaches, railway carriages and aeroplanes. Typical parts for cars are the surrounds of windows and doors, and the stiffener sections which are welded inside the door, boot and bonnet pressings.

The material is presented to the stretch forming machine as a continuous strip of preformed steel or aluminium alloy section. The machine cuts a piece to the required length, grasps and then bends it in several planes, simultaneously stretching the section in order to avoid irregularities in the curved surfaces.

Some sections are formed in quite heavy material, and considerable forces are required to perform the stretching and bending actions. This means that the axes cannot be driven by stepper motors which are commonly used on industrial robots. Instead, hydraulic cylinders must be used.

In order that the finished parts fit accurately into their assemblies, considerable accuracy is required – the current generation of machine calls for positional accuracy of 0.01mm over a total axis length of up to 2m.

To obtain the correct shape, the stretching and bending must take place over up to seven axes simultaneously, all moving in a co-ordinated manner.

The machines must operate quite rapidly, for example machines used on the Ford Mondeo line must produce around 1,000,000 parts per year. They must be extremely reliable because they form part of a just-in-time production system with no parts storage. Downtime of less than 24 hours can stop a major car plant, with thousands of workers laid off.

Overview of the control system of a stretch forming machine

A special purpose servo circuit controls each axis of the machine. This receives positional command signals from a central controller, and compares these with actual position data obtained from an absolute linear transducer. A difference signal is generated and fed to a hydraulic valve that controls the movement of the cylinder that drives the axis.

The central controller is a Virtual Programmable Logic Controller (VPLC) as described in a previous paper to this conference. It takes a bending program expressed as a series of positional values, interpolates them and distributes them to the servo control circuits.

A personal computer (PC) acts as a Human Machine Interface (HMI), allowing bending programs to be entered and adjusted, and machine status to be reported.

Peripheral functions are controlled, including cutting and punching operations. There are interfaces provided to other machines, such as unloading and stacking robots, section forming machines, and welding machines.

The entire control system is programmed in Forth.

The servo controller hardware

The servo controller consists of:

- A single chip microcomputer – the IX1
- A fieldbus interface for connection to the central controller
- A synchronous serial port for connection to the positional feedback device
- An analog output for connection to the hydraulic valve
- Logical inputs and outputs for connection to peripheral devices
- Power supply, circuit protection, and safety circuits

The IX1 single chip microcomputer

This interesting circuit is a hardware Forth engine. It has been described in a previous paper to this conference, but I will briefly summarise its attributes again.

The IX1 is extremely fast and has a single cycle interrupt latency. As a result it is capable of implementing a software serial port at speeds of up to 2.5Mbaud. It is therefore capable of handling the protocols of most of the common fieldbus systems without any hardware assistance.

When used with an Interbus-S fieldbus protocol, which uses a modified shift register serial system running at 500kbaud, there is plenty of spare processing power to carry out other tasks.

Programming the IX1

The manufacturers of the IX1 provide licensed object code to implement the desired fieldbus protocol. An application development system allows user code to be compiled and linked with the protocol code.

The IX has some unusual features and the Forth which it supports does not pretend to be to ANSI standards. Nevertheless it can be easily coded by experienced Forth programmers.

Apart from the slightly disconcerting fact that this is a 12 bit computer (neither of us had encountered one of these since the days of the PDP8!), there was the more startling discovery that words can be defined in any order. This is a somewhat unconventional Forth compiler, which allows forward references only when compiling from a source file, not from direct keyboard entry.

The interface with the protocol code

The application is required to initialise certain parameters for the protocol controller. The baudrate generator and timer interrupt settings must be scaled to match the crystal frequency (normally 16 or 24MHz). The identification codes which the unit will declare to the fieldbus master must be set, along the number of data bytes which the unit will occupy in the shift register structure. Finally, the range of event codes which the application requires the protocol controller to report, and the frequency of reporting, must be set.

```
: APPINIT ( --- ) \ Initialise IBS protocol controller
  $004 BAUD !      \ For 24MHz crystal
  $2EE TIMER !    \ Interrupt rate to match crystal
  $003 IDHIGH !   \ 3 word (6 byte) nodewidth declared
  $03F IDLOW !    \ Analog and digital I/O, no param. Channel
  $006 NODEWIDTH ! \ 6 byte nodewidth used
  $00F EVENT_MASK ! \ All standard events required
  $001 DATACOUNT ! \ Report every event
  640 WATCHDOG !  \ Set watchdog timeout for bus fault
;
```

The main task of the application needs to initialise its own variables as well as the protocol controller. The fieldbus processing can then be started, and events detected and processed. The other main task functions, including the servo control loop, are then processed.

```
: APPLICATION ( --- ) \ The servo controller main task
APPINIT          \ Set up the protocol controller
USERINIT        \ Initialise the servo variables
IBSENABLE       \ Start the protocol controller
BEGIN           \ The main loop
  WOOF          \ The watchdog reset function
  EVENT @       \ Get event flags
  DUP 0 EVENT ! \ Only one event reported at a time
  #NEW_DATA MASK IF \ Valid data cycle reported by protocol
    NEW_DATA_FUNC
  THEN
  .....        \ Other events, e.g. protocol errors
  DROP
  READ_INS SET_OUTS \ Process the logical I/O
  PARAMETERS      \ Deal with virtual parameter channel
  S_FAULTS        \ Service fault conditions
  DO_POSITION     \ Read position transducer
  DO_CONTROL      \ Set the valve position
  DO_SPOOLPOS    \ Check valve spool position
REPEAT
; MAINTASK: APPLICATION \ Set this as the main task
```

Tapping into the interrupt system

The development system allows additional interrupt service routines to be added. For the servo controller, all that is needed is a few more millisecond timers.

```
: INTSCHEDULER ( --- ) \ Main interrupt service routine
  INTFLAG @
  #SWI1 MASK IF SWI1_SERVER THEN    \ Needed!
  #SWI2 MASK IF SWI2_SERVER THEN    \ Needed!
  #TIMER_INT MASK IF TIMEINC        \ Needed!
                                DO_TIMERS    \ Our bit
                                THEN
..... \ Response to other interrupt sources can be added here
  DROP
  INTRET
; INTERRUPT: INTSCHEDULER \ Set this as the interrupt servicer
```

The Interbus data

The servo controller occupies 6 bytes on the Interbus S ring. The input data consists of 3 bytes (24 bits) of command position signal, one byte of logical output image, and one byte each for the address and data of the virtual parameter channel. The output data is very similar, consisting of 3 bytes of actual position, one byte of logical input image, and two bytes of virtual parameter channel.

In addition to the process channel, for input/output data, which is updated every bus cycle, the Interbus-S standard specifies a parameter channel in which less time-critical data can be transferred at a rate of one byte per bus cycle. Unfortunately, the IX protocol does not support the parameter channel. However, parameters are needed by the servo controller, for example, for downloading tuning parameters for the servo control loop. These parameters are held centrally by the control system, so that if a servo control unit ever needs to be replaced, retuning is not required.

The servo controller and VPLC therefore conspire to implement a virtual parameter channel in software.

```
: NEW_DATA_FUNC ( --- ) \ Called when protocol reports good cycle
[ RXBUFFER 3 + ] LITERAL @ 4SL          \ Read command, fiddle to
[ RXBUFFER 4 + ] LITERAL @ 4DROR        \ get 2 x 12 bit words
[ RXBUFFER 5 + ] LITERAL @ 4SL +        \ from 3 x 8 bit bytes
$FFF $7FF 2UMIN COMMAND 2!              \ Constrain & save
[ RXBUFFER 2 + ] LITERAL @ OUTPUTS !    \ Output images
[ RXBUFFER 1 + ] LITERAL @ P_ADDR_IN !  \ Parameter address
RXBUFFER @ P_DATA_IN !                  \ Parameter data
P_ADDR_OUT @ [ TXBUFFER 1 + ] LITERAL ! \ Send parameter address
P_DATA_OUT @ TXBUFFER !                  \ Send parameter data
INPUTS @ [ TXBUFFER 2 + ] LITERAL !     \ Send logical inputs
FEEDBACK 2@                             \ Send 24bits of position,
DUP 4SR [ TXBUFFER 5 + ] LITERAL !     \ fiddle to stuff
$F AND 4DROL [ TXBUFFER 4 + ] LITERAL ! \ 2 x 12 bits
4SR [ TXBUFFER 3 + ] LITERAL !         \ into 3 x 8 bits
FAULTS @ $7FF AND FAULTS !             \ Clear bus inactive flag
;
```

The control loop

There are effectively three control loops for each axis.

The central controller sends out a stream of position command signals (reviewed normally every 5ms). This is open-loop control (no feedback), except that if the actual position deviates from the command position by more than a specified amount, the machine shuts down to avoid collisions between axes. (The consequences of an axis collision are visually similar to the TV series “Robot Wars” but economically different.)

The hydraulic valve has its own control loop in which it attempts to match its own spool position to its command signal, as received from the servo controller. The servo controller monitors the valve spool position, but only for diagnostic purposes.

The servo controller is a pure proportional control loop (no integral or derivative action). A special consideration is that the dynamic characteristics of most axes are dependent on the direction of movement. Therefore, different proportional gain parameters are required in each direction.

A certain amount of fiddling is required to generate a 12 bit valve position signal from 24 bit command and feedback systems.

```
: PROP_LOOP ( --- ) \ Proportional control loop
  CALC_ERR           \ Slope limit command and calc error
  DUP >R             \ Preserve sign of result
  DABS               \ Remove sign for calculations
  R@ $7FF U> IF      \ Fetch gain for appropriate direction
    GAIN_RT           \ Obtained from the parameter channel
  ELSE
    GAIN_EX
  THEN @
  UMD*              \ err * gain
  DUP $7FF U> IF     \ Set max. value
    2DROP $FFF $7FF \ Effectively, 2UMIN
  THEN
  10 DIVU NIP       \ /10
  $7FF UMIN         \ Ensure max value not exceeded
  R> $7FF U> IF NEGATE THEN \ Restore sign
  VALVE !          \ Save result
;
```

Other interesting bits

The feedback device provides a 24 bit gray code, which must be converted to binary before use. The implementation given below works, but we are not sure it is optimal. We include it because it is the tradition of this conference that certain delegates gain enjoyment from reducing such code segments to their most elegant and simple forms.

```
: GRAY>BINARY ( g---b ) \ Convert 24bit gray code to 24bit binary
GRAY 2! 0 $800 BMASK 2! \ Init conversion locations
GRAY 2@ BMASK 2@ 2AND BINCOD 2!
23 ?FOR
  BMASK 2@ BINCOD 2@ 2AND UD2/ \ bin msb -> msb-1
  BMASK 2@ UD2/ 2DUP BMASK 2! \ Shift mask
  GRAY 2@ 2AND \ gray msb-1
  2XOR BINCOD 2@ 2OR BINCOD 2!
NEXT
BINCOD 2@
;
```

Uses of the servo controller

The new servo device is currently used in building the following cars:

Ford Mondeo new model (6 parts)
GM Corsa new model (2 parts)
GM Vectra new model (2 parts)
Citroen new model (unnamed) (2 parts)

Previous versions of stretch forming machines, in which the central control system and the HMI are programmed in Forth, but where the servo controller is a pure hardware implementation, are used in building the following cars:

Ford Focus (4 parts) (Also made in Detroit USA)
Ford Australia (various names) (2 parts)
Land Rover Freelander (12 parts)
GM Astra (4 parts)
VW Tico (2 parts) (not sold in the UK)
Rover 200 (2 parts)
Rover 400 (4 parts)
Honda Civic (4 parts)
Honda Accord (4 parts)
Saab 9000 (2 parts)
Jaguar XJ400 (1 part)
Jaguar S type (4 parts)
Ford Mondeo (old model) (now spares only) (4 parts)

Numerous other cars have parts made using control systems that are partly controlled by Forth.

Advantages of using the IX1

1. The applications programmer uses only one language. Forth is used for all parts of the control system.
2. The computing power available in the IX1 means that a single chip is able to act as both fieldbus interface and loop controller.
3. Placing an intelligent device in the servo controller has greatly eased the tuning of axes.
4. The use of a fieldbus has greatly simplified the machine wiring.
5. The control system is now almost all digital, resulting in much greater accuracy. The only remaining analog signal is the valve command.

Conclusion

The use of the Forth engine IX1 in a fieldbus servo controller has resulted in a highly effective and simplified control system for an extremely complex operation.

References

1. The stretch forming machines are made by Travers Metal Products Ltd., Mushet Industrial Park, Coleford, Gloucestershire, GL16 8RD. Tel. (+44) 1594 810343
2. The IX1 microprocessor is made by M2C, Schauenburger Strasse 15-21, D-20095 Hamburg. Tel. (+49) 40 325 682 0
3. "IX: The first Forth single chip controller", K. Schleisiek-Kern, EuroFORTH 1994