ANL/PPRNT--89-104

DE89 008402

DISCLAIMER

Charles E. Cohn
445 Ridge Avenue
Clarendon Hills, IL 60514
(312)323-6695

PROGRAMMING FOR A NUCLEAR REACTOR INSTRUMENT SIMULATOR

(Work performed at Argonne National Laboratory under the auspices of the U. S. Department of Energy)

A new computerized control system for a transient test reactor incorporates a simulator for pre-operational testing of control programs. The part of the simulator pertinent to the discussion here consists of two microprocessors. An 8086/8087 reactor simulator calculates simulated reactor power by solving the reactor kinetics equations. An 8086 instrument simulator takes the most recent power value developed by the reactor simulator and simulates the appropriate reading on each of the eleven reactor instruments. Since the system is required to run on a one millisecond cycle, careful programming was required to take care of all eleven instruments in that short time. This note describes the special programming techniques used to attain the needed performance.

Each instrument comprises a neutron detector that produces a current proportional to the incident neutron flux and a linear or logarithmic current amplifier, as shown in Figure 1. The system responds to reactor powers in the range of 50 watts to 20,000 megawatts, corresponding to detector current ranges of 10^{-11} to 10^{-3} amperes. A 12-bit ADC digitizes the output of the current amplifier for use by the control computer.

The simulation is done by introduction of a current into the amplifier input. This current is generated by application of 0 to 10 volts from a 12-bit DAC through any one of nine series resistors going by decades from 10^4 to 10^{12} ohms. The simulator processor generates the DAC datum and selects the appropriate resistor by activating the driver that energizes the corresponding reed relay.

The simulator receives each power value, in units of megawatts, as an 8087 short real datum, in which the MSB is the sign, the next eight bits are the biased exponent, and the least significant 23 bits form the normalized fraction, with the most-significant one-bit dropped. The process uses two tables, indexed by the exponent, for each instrument. The first contains bytes that select the appropriate series resistor for each value of the exponent, while the second contains words by which the fraction is multiplied to form the DAC datum.

As shown in Listing 1, the processor first separates out the exponent, extends it to 16 bits with leading zeros, limits it to the appropriate range, and places it in the SI register for indexing. It then takes the high-order 15 bits of the fraction, merges in the missing MSB, and places the result in the CX register for use as a multiplier.

For each instrument, then, the appropriate resistor selection byte is extracted from the table and put out to the instrument, as shown in Listing 2. Then the SI register is multiplied by 2 through a left shift to prepare it for indexing words. The fraction is multiplied (unsigned) by the appropriate table entry for each instrument, and the product is put out to the DAC as shown in Listing 3.

In the actual program, the DAC values are not put out as soon as they are calculated, but are stored temporarily in assigned memory locations and then put out in sequence after all the multiplications are done. The resulting delay between resistor selection and datum output helps to avoid switching transients when the resistor is changed. The relay drivers were designed to have fast dropout and slow pullin, so that the previously-selected resistor will be disconnected by the time the datum is output, but the new resistor will not yet be connected. The entire switching operation takes a few tenths of a millisecond, and the system timing is such that the ADCs will never be strobed during switching.

The tables are prepared as shown in Listing 4, which is written in Intel Fortran-86, using instrument calibration factors obtained from preoperational tests. It selects a resistor for each exponent such that the largest DAC output possible with that exponent will be greater than 1 volt, or one-tenth full scale. (Note that the cutoff point is made slightly less than 1 so that no DAC datum larger than 4095 can be generated. Also, the datum is scaled to 8192 instead of 4096 to allow for rounding by adding 1 and shifting right.)

The system has worked well during extensive testing. The main difficulty experienced is switching transients caused by capacitive coupling between the coils and contacts of the reed relays. This effect limits the usefulness of the method for currents less than 10^{-8} amperes.

Figure Caption:

Figure 1. Block diagram of typical nuclear reactor instrument with provisions for simulation

Listing 1: Processing common to all instruments

```
        MOV     AX,WORD PTR POWER[2]    Get hi-order word with exponent.
        ROL     AX,1                    Discard and ignore sign bit.
        MOV     AL,AH                   Right-adjust exponent in AX.
        MOV     AH,0
        SUB     AX,110                  Subtract minimum exponent.
        JNS     NOTLOW                  Is exponent below minimum?
        MOV     SI,0                    If so, set up lowest possible
        MOV     CX,8000H                power as result.
        JMP     READY
NOTLOW: CMP     AX,29                   Is exponent above maximum?
        JBE     NOTHIGH                 (RANGE = max. exp. - min. exp.)
        MOV     SI,29                   If so, set up highest possible
        MOV     CX,0FFFFH               power as result.
        JMP     READY
NOTHIGH: MOV    SI,AX                   Put exponent in index register.
        MOV     CX,WORD PTR POWER[1]    Get high-order part of fraction.
        OR      CX,8000H                Replace assumed high-order 1-bit.
READY:                                  Fraction now in CX register.
```

Listing 2: Selection of simulation resistor for each instrument

```
MOV     AL,RESTAB[SI]       Resistor selec. byte from table.
OUT     PORT,AL             Send to instrument.
```

Listing 3: Output to DAC for each instrument

```
MOV     AX,CX               Retrieve fraction.
MUL     SCALTAB[SI]         Mult. by scale factor from table.
ADD     DX,1                Round
SAR     DX,1                and rescale.
MOV     AX,DX               Send to instrument.
OUT     PORT2,AX
```

Listing 4: Preoperational formation of resistor selection and scale factor tables

```
C - The following coding initializes the REAL*4 variable POWER to a
C   positive floating-point number with the largest possible fraction
C   and exponent equal to the smallest exponent within range
C   ( = 110 in this example)
          DIMENSION IPOW(2)
          EQUIVALENCE(POWER,IPOW)
          DATA IPOW(1),IPOW(2)/-129,14207/
C - (IPOW(2) initialized to 128 x lowest exponent + 127)
          I = 0
C - Start of loop to step through exponent values for POWER
     1    I = I + 1
C - In the actual program, the following coding is enclosed in a DO loop
C   over all instruments.
C
C - Calculate instrument input current equivalent to power value
C   (CALFAC = detector calibration factor, amperes/megawatt)
          CURRENT = POWER * CALFAC
```

```
C - Starting with the smallest simulation resistor (10^4 ohms),
C   loop until a resistor is found
C   which requires an input voltage over 0.99987792 (= 8190.9999/8192)
C   (i.e. 1/10 scale)
C   to produce the current
C   or until the largest resistor is selected.
C   (MAXRES is the total number of resistors)
          VOLTAGE = CURRENT * 10000.
          IRESIST = 1
    2     IF (VOLTAGE .LT. 0.99987792 .AND. IRESIST. LT .MAXRES) THEN
          VOLTAGE = VOLTAGE * 10.
          IRESIST = IRESIST + 1
          GO TO 2
C - Obtain multiplier and resistor selection code
C   (MULTPR = multiplier, RESCOD(I) = resistor selection code for exponent I,
C   RESBYT = array of selection codes corresponding to resistors
          MULTPR(I) = NINT (VOLTAGE * 819.2)
          RESCOD(I) = RESBYT(IRESIST)
C - End of coding that is looped over all instruments
C
C - If less than 30 exponent values have been treated,
C   increase exponent by 1 and repeat
          IPOW(2) = IPOW(2) + 128
          IF (I .LT. 30) GO TO 1
```

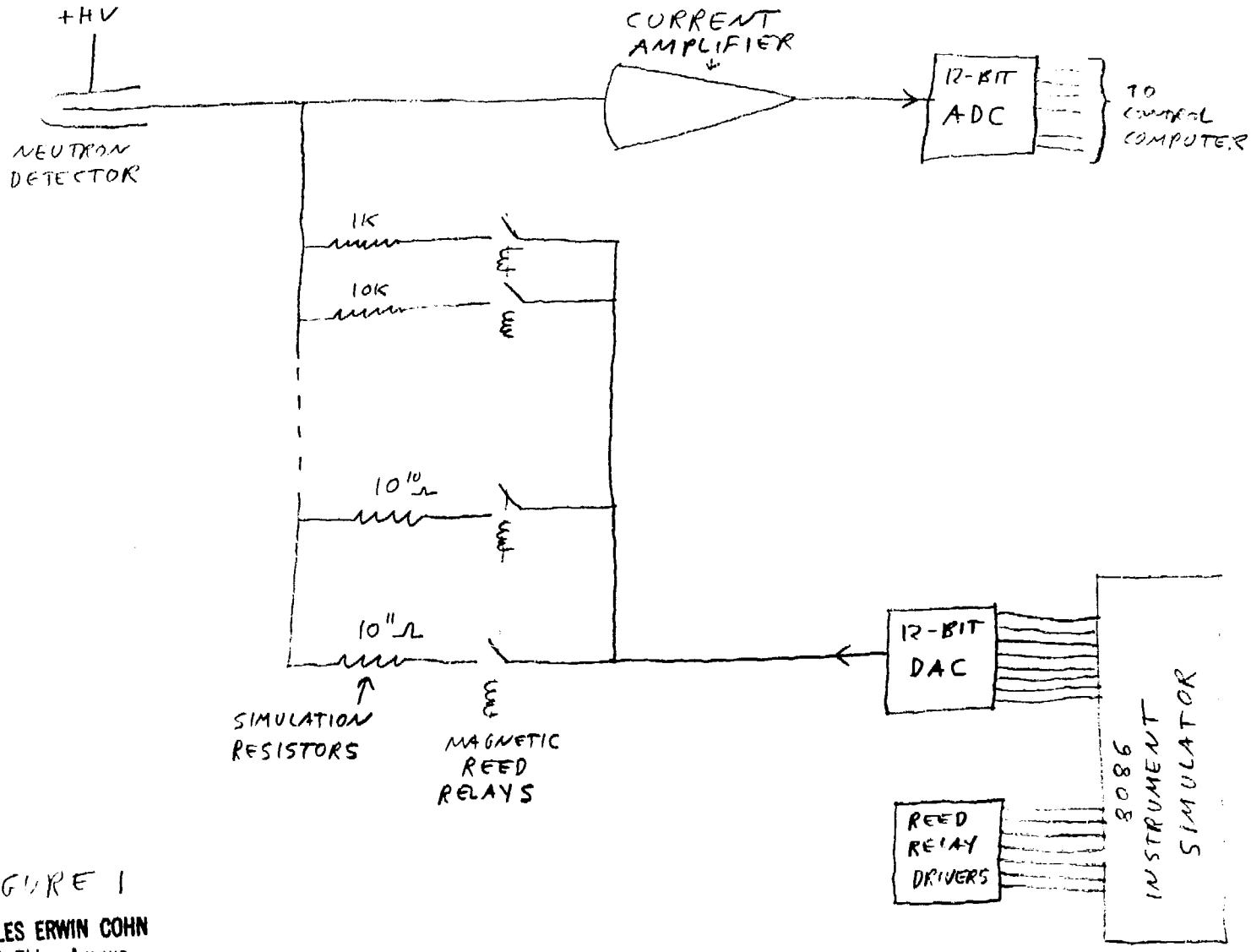
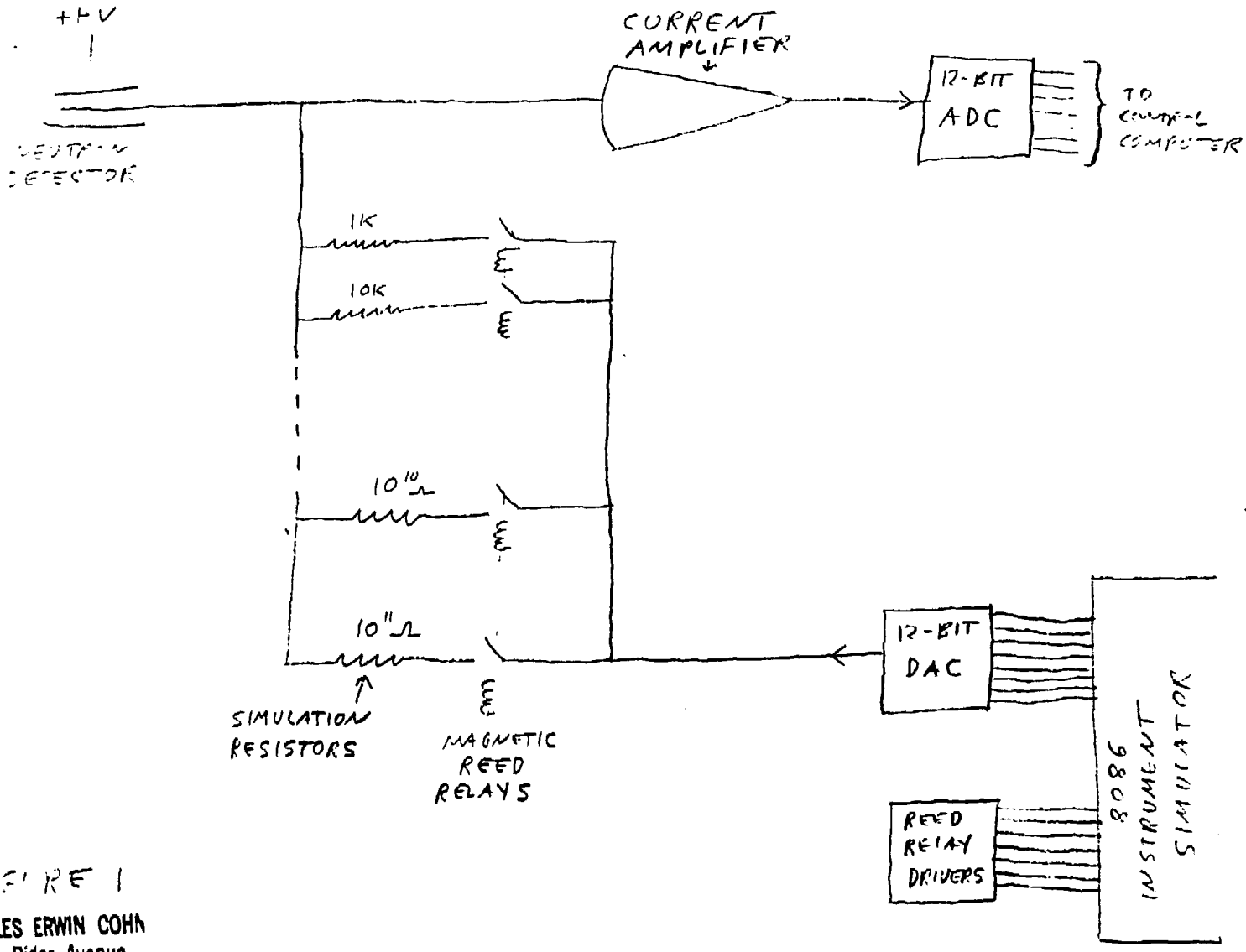


FIGURE 1

CHARLES ERWIN COHN
445 Ridge Avenue
Clarendon Hills, Ill. 60514



= FIRE 1

CHARLES ERWIN COHN
445 Ridge Avenue