RC-3 Relay Computer
Theory of Operation

Revision 1
13 Jan 2013

Phil Ryals

Table of Contents

Preface

The RC-3 Relay Computer is based on the design of a similar relay computer created by Dr. Harry Porter, a computer science professor at Portland State University in Oregon. Many details concerning how that design was developed, and how the various circuits work, are contained in his paper available at http://web.cecs.pdx.edu/~harry/Relay/RelayPaper.pdf, which is the best place to start to understand the RC-3.

Note

To avoid confusion, places where this design differs significantly from the Harry Porter relay computer (referred to herein as *HPRC*) will be called out in this document.

This document attempts to explain RC-3 operation at a level suitable for individuals who may be called upon to maintain and repair the computer, as well as those seeking a greater understanding of how it works. The reader is assumed to have access to a complete RC-3 document repository. Descriptions in this document will refer to schematics, timing charts, and other drawings contained in the RC-3 design documents.

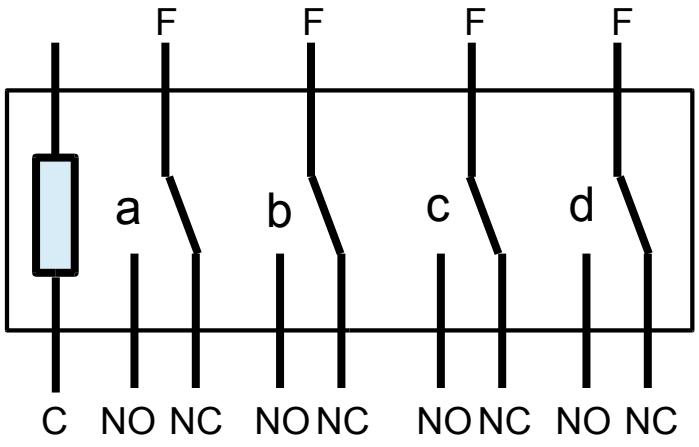
Note

Drawings will be referred to by their file name in the document repository. For example: *Sequencer1-x*. The number before the hyphen, if any, is the page number for a multipage drawing. The number after the hyphen, represented by the generic *x*, is the revision level of that drawing. The current revision level of all documents in the repository can be determined from the *DocumentControl* file contained therein. *GeneralNotes1-x* may be useful in interpreting the schematics.

After an initial design overview, each functional section of the RC-3 is explained in some detail, with references to applicable drawings and/or timing charts. Functional sections are defined as groupings of circuits that are closely interrelated, which can be operated and debugged with the remainder of the computer disconnected from them.

Note

By convention, relay contacts are referred to by position (a, b, c, d), where contact set a is closest to the coil (regardless of the relay orientation on the schematic), and by contact within that set: Normally Open (NO), Normally Closed (NC), and Flap (F). The powered coil connection is referred to as Coil (C). Although not shown, the other side of the coil is always grounded.



Design Overview

Architecture

RC-3 is an 8-bit computer, with a 16-bit address bus. Refer to the *SystemArch* drawing for details. It has the following registers:

A – general purpose register

B – general purpose register, base input to the ALU

C – general purpose register, additional input to the ALU

D – general purpose register

M1/M2 – memory address register, used on the 16-bit bus as **M** in load/store operations; M1/M2 may be used on the 8-bit bus as general purpose registers

X/Y – general purpose 8/16-bit register, used as **XY** on the 16-bit bus to save the return address for subroutine calls

J1/J2 – jump register, used as J on the 16-bit bus as the target address for jumps

INST – instruction register, used during decoding of instructions

PC – program counter

INC – increment register, used to increment the Program Counter

COND – condition code register, stores results of tests on ALU operations

Memory

RC-3's 16-bit address bus allows access to 64 Kbytes of memory. This is divided into a 32-KB static RAM in the lower half of the address space, and a 32-KB EEPROM in the upper half. Test, demo, and other programs can be written into the EEPROM using an external PROM burner.

Arithmetic Logic Unit

The ALU receives its inputs from the B and C registers and provides, via an adder section and a logic section, the following operations:

ADD B+C

INC B+1

AND logical AND of B with C

OR logical OR of B with C

XOR logical exclusive OR of B with C

NOT bitwise complement of B

SHL B circularly shifted left by one bit

The C register is ignored by the ALU for the INC, NOT, and SHL operations, which operate only on the contents of the B register.

Condition Codes

Results of all ALU operations are tested for Sign (most significant bit a 1), Zero (all bits 0), and Carry (carry out of most significant bit of the adder for ADD and INC instructions). Results of these tests are stored in a Condition Code register that is used by the GoTo operation and its variants to control conditional branching. For programming convenience, a Non-Zero (one or more bits not 0) value is also made available to the conditional branch instructions.

Instruction Set

The instruction set is documented completely in *RC-3InstructionSet-x* and is laid out by op-code in *RC3InstructionSetMapR3*. There are a number of differences between the RC-3 design and the HPRC design:

ALU ALU function codes for RC-3 are incremented by one, making 000 the unused function code. All others are one greater than in the HPRC instruction set.

LOAD duplicate op-codes 94-97 have been made no-ops.

STORE duplicate op-codes 9C-9F have been made no-ops.

MOV16 source register code 11 now loads contents of the front panel address switches and no longer is a Halt.

MOV16 duplicate op-codes A8-AB have been made no-ops.

LDSW a new instruction to load the A or D register from the front panel data entry switches has been added.

HALT the Halt instruction no longer clears the PC; it is left at the current location+1.

HLTRL a new instruction has been added to load the PC from the front panel address switches, then Halt.

INCXY duplicate op-codes B2-BF have been made no-ops.

PRINT op-code B1, formerly a duplicate of INCXY, has been remapped to the Print operation.

GOTO behavior of the c (carry) bit has been inverted; the PC is loaded if the carry bit is set instead of if it is cleared.

Instruction Decoding & Execution

Instructions are decoded by sequential relay logic (described below in the Control section of this document). Instructions are executed by a series of pulses produced by the Sequencer (also described below) gated to the appropriate subsections (registers, ALU, etc.) by the decoding relays in the Control section. Each pulse activates a simple operation, such as gating a register onto a bus, selecting an ALU output, etc. A total of 19 such pulses, produced at different times and with varying widths, are required to perform the complete set of RC-3 instructions.

Timing & Pulse Generation

Timing for RC-3 is provided by the Clock section that produces a roughly square clock signal at 3 Hertz, which is fed to a 24-stage Sequencer. Pulses are produced by the Sequencer on both positive and negative excursions of the clock signal, so the operating speed of the computer is 6 Hz (there are six clock periods per second). Instructions take from 8 to 24 clock periods. Each stage of the Sequencer produces two pulses: one 1 clock period long, and another 2 clock periods long. These are combined in various ways to generate the 19 pulses required for machine operation.

Front Panel Operations

A feature added to RC-3 (not present in the HPRC design) is the convenience of front panel operation switches. Operators need not remember the sequence of control pulses required to perform basic operations; instead they can be performed by single switch toggles on the front panel. These functions are available:

Load Address loads the PC with the contents of the address entry switches.

Examine reads the memory location pointed to by the PC and stores it in the A register.

Examine Next performs an Examine, then increments the PC (useful for reading out a program or data table in memory).

Deposit writes the contents of the data entry switches into the memory location pointed to by the PC.

Deposit Next performs a Deposit, then increments the PC (useful for writing a program sequentially into memory).

Restart releases a program-generated Halt (this required turning Clock power off in the HPRC).

Momentary (spring-loaded) switches are used for all these operations. Pulses needed to perform front panel operations are produced by a separate Auxiliary Clock (not in HPRC), which is triggered by the switch operation.

Other normal operation switches are provided as in the HPRC:

Reset initializes the Sequencer.

Run/Stop controls Clock signal to the Sequencer.

Clock Step allows single stepping the Sequencer when the machine is stopped.

Debugging

Indicators are provided for observing contents of registers and machine state, and switches are provided to manually generate control signals for all sections of the machine. In the HPRC design these debugging switches were always active, so had to be in the correct (non-energized) position to allow normal machine operation. In RC-3 those switches not needed for routine operations (the set described above in Front Panel Operations) may be disabled by a key switch that controls power to them. Only authorized users with a key can activate the debugging switches. Since there are some debug switch settings that can physically damage relay circuitry, this provides a level of protection against unskilled operators.

Clock, Sequencer, and Pulse Distribution

The Clock and Sequencer, along with the Auxiliary Clock (described in its own section, below) produce all the pulses used to execute the various operations of the machine.

Clock

The clock circuit is shown on drawing *Clock-x*. Relays K101–K104 form a four-stage ring counter with feedback from the fourth stage to the first stage, operating as a free running oscillator. Capacitors C101–C104 provide a delay at each stage. When a relay coil is energized, the capacitor connected to it is charged. When the driving voltage is removed, the capacitor discharges through the coil and keeps the relay closed for longer than it would have been otherwise. This establishes the basic clock rate for the computer, approximately 167 ms per clock tick.

Parallel resistors R101–R104 increase the discharge rate of the capacitors slightly, and were selected to produce the desired clock rate. Resistors R105–R108 are used to limit the in-rush current into the capacitors to below the 5A rating of the relay contacts.

The clock starts when power is applied to the circuit. Initially, all relays are de-energized and open. The coil for relay K101 then receives power through contacts K102dNC and K103cNC. When K101 closes, it applies power to K102. K102 applies power to K103 and removes power to K101. This process repeats as each relay powers the one in front of it and removes power from the one behind it. The timing relationship among the four relays, and the conditions that energize each one are shown on *ClockTiming-x*. Exactly two relays are energized at any given point in time (after the initial start).

The oscillator's output signal (CL') is produced by an exclusive OR of K103 and K104 via their b contact sets; that is, when K103 and K104 are in different states (one closed, one open), the clock is high, and when K103 and K104 are in the same state (both closed, or both open), the clock is low, as shown in the timing diagram.

K105 is the clock freeze relay. When set by receipt of a Halt signal from the Control section (execution of HALT or HLTRL) K105 provides a lockup voltage to relays K101 and K103. This causes the clock to stop with CL' low in either T2 or T4 as shown on the clock timing diagram. K105 itself locks up on power provided through S703 and J701. S703 is a spring-loaded front panel switch labeled Restart. When opened, K105 drops out removing the lockup voltage to K101/K103 and the clock starts running again. J701 is provided for an external normally-closed switch intended to be used by the public to restart demo programs.

Power for the clock circuit is controlled by S701, which may be used to shut the clock off during debugging operations. S704 is the Run/Stop switch. In the Stop position it energizes the freeze rail, which stops the clock. It also provides power to S702 allowing the clock to be single stepped when the freeze relay is not set. S704 also controls power to other front panel operations switches, making them active only when the machine is in Stop mode.

D701 prevents power back feeding from K105 to S702 and FP-OPS-POWER. D702 prevents power back feeding from the clock signal to FP-OPS-POWER should S702 have been left closed. I701–I704 indicate the state of the clock relays. I705 shows the clock signal, and I706 shows when the freeze relay is set. C113 is a power supply bypass capacitor.

Sequencer

The sequencer is detailed on drawings *Sequencer1-x* through *Sequencer3-x*. Relays K201–K224 compose a 24 stage ring counter, with feedback. Relays K225–K227 are clock repeaters; they are driven by the Clock signal and feed pulses to the ring relays on both clock high and clock low states. Relays K228–K231 are abort relays (described below) used to shorten the pulse sequence from 24 steps to 8, 10, 12, or 14 steps, as needed by the instruction being executed.

Program execution is begun by the operator manually closing S840, the front panel Reset switch. Note that S840 is fed by FP-OPS-POWER, which is available only when the Run/Stop switch is in the Stop position. Once K201 is closed, it locks up on the voltage it receives via K201dNO from K203aNC. Moving Run/Stop to the Run position then starts the clock. When the clock first goes high, K225aNO energizes K201cNO, which sets K202. K202 locks up on the voltage it receives through K202dNO from K204aNC.

When the clock switches to low, K225 drops out driving K225aNC high. This sets K203 through K202cNO. The closing of K203 removes voltage from K203aNC, which is holding K201 closed, and K201 drops out. This process repeats through successive stages: a relay closes on the next clock transition (either high or low), which sets the relay following it, and releases the relay two positions behind it. Note that K223 and K224 are connected just like the other relays in the chain, back to K201 via the ~S1 and ~RESET signals. If no abort relays are set, the chain continues through the normally closed d contact on each abort relay, and the sequencer runs through all 24 stages, then back to the first stage and the process repeats as long as the clock is running.

Note that each relay stays closed for two clock ticks (until the relay in the second stage following it closes), so all S1–S23 signals are two clock periods long. Each of the S0'–S23' signals goes high for only one clock tick. These signals are fed to the Pulse Distribution Section for display and generation of the 19 pulses required for instruction execution. The timing relationship among the sequencer pulses is shown in *SequencerTiming1-x* through *SequencerTiming3-x* for the full 24 stage sequence.

Most instruction op-codes require fewer pulses for execution than the full set of 24. When an instruction is decoded it may require only 8, 10, 12, or 14 clock periods to execute. If this is the case, a signal is produced in clock period 5 that sets the appropriate Abort relay, which locks up on the ~S1 voltage from K202aNC (indicating that K202 is not set). When the sequence reaches the abort relay, instead of continuing down the chain it drives the RESET line forcing K201 to set just as it would have following K224 if the full sequence had run. When K202 is next set, ~S1 goes low, releasing the abort relay. Timing of the abort sequences is shown in *SequencerTiming4-x* and *SequencerTiming5-x*.

C201 and C202 are power supply bypass capacitors.

Pulse Distribution

Integral to the sequencer, but diagramed in a separate set of drawings (*PulseDistribution1-x* through *PulseDistribution4-x*), are the pulse distribution and display circuits. All Sn' signals are routed to indicators for display, as shown on *PulseDistribution1-x*. Note that S0' indicates K201 is closed, and so on up to S23', which indicates that K224 is closed.

As mentioned earlier, a set of 19 different pulses is required for instruction execution. These distinct pulses start at varying times and are from 1 to 3 clock periods long. The complete set of pulses, showing their start times and durations, is shown on drawing *MasterTiming1-x*. Annotations on the drawing show the general use for the pulses, along with the points in time at which the sequencer may be reset. More detailed indications for how the pulses are used will be covered in the Instruction Execution section of this document.

The sequencer stages directly produce all required one and two time period long pulses. Where three time period long pulses are required, these are produced by ORing two adjacent two time period long pulses together using diodes to prevent signal back feeding from one pulse source to the other. Because the two adjacent 2-tick pulses overlap for one clock tick, there is no possibility of a signal glitch where they are joined.

Pulse generation and display is diagrammed on *PulseDistribution2-x* and *PulseDistribution3-x*. I748–I766 display the generated pulses, while switches S711–S729 may be used to force a particular pulse for debugging purposes. Note that all these switches receive power from V_d, so they are active only when the Debug switch is on. Sequencer sources for the pulses are shown at the top of the drawings, and they are distributed to the Control section via P202.

D232 and D233 OR S1 and S2 together to produce the 3-tick long P-A. Likewise, D234 and D235 OR together S4 and S5 to produce P-E, D236 and D237 OR S8 and S9 to produce P-J, and similarly D238 and D239 OR S15 and S16 to produce P-N. These pulses are all three clock ticks long.

The remainder of the required pulses are either 1-tick long, produced by an Sn' signal from the sequencer, or are 2-ticks long, produced by an Sn sequencer signal. Note that not all Sn pulses are actually used. Unused signals are shown on the sequencer drawings by grayed-out labels.

The final sheet of pulse distribution drawings, *PulseDistribution4-x*, provides for display of the state of the abort relays, and switches to manually set them. Those switches are also only active when the Debug switch is on.

Buses

Data moves around the computer on two parallel buses: the 8-bit data bus, and the 16-bit address bus. The buses actually consist of 24 wires in interconnection cables that daisy chain through all the units requiring access to either 8-bit data or 16-bit addresses. They have associated with them a set of display indicators and a set of entry switches, separate for each bus.

Data Bus

Indicators I1301–I1308, on *DataBus1-x*, display information present on the 8-bit data bus. *DataBus2-x* shows the data entry switches and their gating relays. S810–S817 may be used to input data into the machine. These switches do not require Debug mode to be active.

Relays K617–K618 are used to gate the contents of the data entry switches onto the data bus, either under program control via the SEL-DATA-SW signal produced by the LDSW instruction, or under manual control by closing S837 when Debug is on. Front panel

Deposit and Deposit Next operations also produce a SEL-DATA-SW signal to read the data entry switches onto the bus. I1328 indicates when the switches are selected onto the bus.

Note

In the HPRC design, the data entry switches sat directly on the data bus and there were no gating relays. The switches had to be set to all open to allow normal machine operation, and there was no provision for data entry under program control.

Address Bus

Indicators I1309–I1324, on *AddressBus1-x*, display addresses present on the 16-bit address bus. *AddressBus2-x* shows the address entry switches and their gating relays. S818–S833 may be used to input addresses into the machine. These switches do not require Debug mode to be active.

Relays K619–K622 are used to gate the contents of the address entry switches onto the address bus, either under program control via the SEL-ADDR-SW signal produced by the HLTRL instruction, manually using the Load Address front panel operation (it also produces a SEL-ADDR-SW signal), or manually by closing S838 when Debug is on. I1329 indicates when the switches are selected onto the bus.

Note

In the HPRC design, the address entry switches sat directly on the address bus and there were no gating relays. The switches had to be set to all open to allow normal machine operation, and there was no provision for address entry under program control.

Memory

The memory section contains a 32 Kbyte static RAM chip (at hex addresses 0000–7FFF) and a 32 Kbyte EEPROM chip (at hex addresses 8000–FFFF). Programs and data may be stored in the EEPROM using an external PROM burner, or they may be entered into SRAM via the Deposit or Deposit Next front panel operations. Relays are used to convert from the 12V logic level used in the relay computer to the 5V logic level used by the chips, and vice versa.

Note

The HPRC design did not include the 32 Kbyte EEPROM chip. All programs had to be manually entered into RAM using front panel switches.

Memory1-x shows the data bus to memory interface. Relays K601–K608 sit directly on the 8-bit data bus and track its state. Contact set d on each relay switches between 5V for a ONE and ground for a ZERO on that bit of the data bus. When a memory write operation is required, the Control section generates a bus-to-memory signal (BTM) that closes K623 and K624, gating the data bus value onto the 5V memory data bus. (M7–M0).

Memory2-x shows the interface between 5V memory logic and the 12V relay data bus. The memory output bus (MO7–MO0) is driven by a driver chip that pulls each pin to ground when that bit is a ONE. Memory read operations energize the MR signal causing K625 and K626 to connect the memory output bus to the coils of K609–K616. Each ONE bit causes its associated relay to close, connecting 12V to the aNO contact, which drives the main data bus.

S839 may be used to remove voltage to the interface relay coils if needed for debugging operations. I911 indicates when memory is enabled. C605 is a power supply bypass capacitor.

Note

This is the only place in the entire design where relay coils do not have one end grounded. K609–K616 have one end of their coils bused to 12V through the memory enable switch. The MOn signals then pull the other end of the coil to ground if the relay is to be energized.

The address bus to memory interface is shown on *Memory3-x* and *Memory4-x*. K642–K656 convert the 12V logic of bits A14–A0 of the address bus to 5V logic for the memory address bus. Relay K641 enables either the SRAM chip or the EEPROM chip, depending on the state of bit A15 (determines which half of the memory address space is being requested). The #OE (output enable) signal comes from memory control on *Memory5-x* and gets gated by K641 contact set a to either the RAM (#OER) or EEPROM (#OEP). K641 contact sets c and d connect the active chip select signal (#CSR for RAM, #CSP for EEPROM) to ground and the inactive one to +5V. These signals are all active when low, hence the # prefix. C602 is a bypass capacitor on the 5V power line.

Memory5-x shows the memory control circuits. When a memory write is requested by the MEM-WRITE signal being asserted by the Control section, K627 closes, connecting the write enable signal #WE (active when low) to ground. When a memory read is requested by the MEM-READ signal from Control, K628 closes, connecting the output enable signal #OE (active when low) to ground. Contact set a of K628 disconnects the bus-to-memory signal to the data bus to memory interface relays whenever a memory read is in progress. I1327–1325 indicate the state of the memory control signals and S834–S836 may be used to force those signals for memory debugging whenever the machine is in Debug mode.

Memory6-x shows the wiring of the memory chips themselves. U601 is a 28C256 32 Kbyte electrically erasable programmable read-only memory (EEPROM) chip and U602 is a 62256 static RAM chip. These devices have tri-state output buses that emit high (+5V) or low (0V) signals only when their output enable pin is pulled low. When the output enable pin is high, the data output pins go into a high impedance state, so do not disrupt the operations of other devices on the output bus.

C606–C608 are high frequency noise bypass capacitors on the 5V power connection of each chip. C603 is a bulk power bypass capacitor. IM601 is an 8-bit LED display module that shows the state of the 5V memory data bus. IM602 and IM603 are two 8-bit LED display modules that indicate the state of the 5V memory address bus.

R601 is a pull-up resistor that disables modification of the EEPROM contents. R602 pulls up the EEPROM output enable to make sure it doesn't go low during switching transients occurring on the relay interface. The combination of R605 and the series LED pulls up the chip select signal likewise to make sure it doesn't go low during relay switching transients. When its chip select is low, the LED is on to indicate selection of the EEPROM.

In a similar fashion, R603 pulls up the write enable pin of the SRAM, R604 pulls up the output enable pin, and R606 plus the series LED pulls up the chip select pin. These pull up resistors all protect against any switching transients from the interface relays. The LED shows when the SRAM has been selected.

U603 is a ULN2803 octal high current Darlington transistor array. A high input on any of its eight input pins pulls the corresponding output pin to ground. These eight signals constitute the memory output bus (MOn), and are used to drive the 5V to 12V conversion relays for memory output data (drawing *Memory2-x*).

Registers

Registers are used for temporary storage of data and addresses during machine operations. All registers used in RC-3 are of the same basic design, with some variations depending on whether a particular register is 8-bit, 16-bit, or 8/16-bit, and how it is used.

Register Usage

Refer to the System Architecture (*SystemArch*) drawing to see how the registers are connected to the data and address buses. Note that the arrowheads indicate directions of allowed data flow.

Registers A, B, C, D, X, Y, M1, and M2 may be used as general purpose 8-bit registers on the data bus.

The 16-bit Register XY, a concatenation of X and Y, may be used to do address arithmetic on the 16-bit address bus. It is also used by the Call instruction to store the return address for a subroutine call.

The 16-bit Register M, or memory register, a concatenation of M1 and M2, is used as a memory address for the Load and Store operations. It is loaded a byte at a time from memory using the 8-bit data bus (as M1 and M2), then accessed as an address on the 16-bit bus. It cannot be loaded from the address bus.

The 16-bit Register J, or jump register, a concatenation of J1 and J2, is used as the target address for various branching operations. It is loaded a byte at a time from the 8-bit data bus (as J1 and J2), then accessed as an address on the 16-bit bus. It cannot be loaded from the address bus.

The Instruction Register (Register INST) is loaded with instruction op-codes from the 8-bit data bus during program fetch operations. Its output is fed directly to the Control section for instruction decoding.

The Program Counter (Register PC) is a special purpose 16-bit register used as an address for fetching instructions and their operands from memory. It is normally incremented after each byte is fetched.

The Increment Register (Register INC) is used during incrementing of the Program Counter. It is loaded with the value of the address bus + 1 directly from the Incrementer, then accessed via the 16-bit address bus to reload the PC with PC+1.

The Condition Code Register, a part of the Arithmetic Logic Unit, stores the Sign, Carry, and Zero states from the current ALU operation for use by conditional branching instructions. It is discussed in this section because it is designed similar to the other registers.

Registers A–D

Register A is typical of the basic register design. Refer to *RegisterA-x* for this description. Relays K401–K408 are the bit storage relays. K409 and K410 are gating relays that connect the register to the data bus. K411 and K412 are the load/select control relays, described below.

Note that if a bit relay becomes set by a voltage coming in off the bus, it will lock up on the voltage coming from K411aNC, latching the ONE value received from the bus. This rail, feeding all bit relays, is called the hold line. If voltage on the hold line is removed while the register is disconnected from the bus, all bit relays will drop to their open ZERO state.

When the register is idle (no voltage applied to either the LD-A or SEL-A inputs), K412 is closed on the ~LD (not load) signal it receives from K412bNC. With K412 closed, the ENABLE line is not driven and the gating relays are open, isolating the register from the data bus.

To place a value stored in the register onto the data bus, the Control section asserts SEL-A (Select-A). This activates the gating relays, which place the contents of the register on the data bus.

To load the register with a value gated onto the data bus from another location (register, memory, or ALU), Control asserts LD-A (Load-A). LD-A closes K411, which drops the voltage on the hold line and all bit relays open to their ZERO state. K411 closing also drops the voltage on the ~LD line. This causes K412 to drop out at the same time as the bit relays. Once K412 opens, it applies voltage to K412aNC. A short time later the gating relays close and any voltages on data bus lines are applied to the coils of the bit relays. The time delay between dropping of the hold line and closing of the gating relays ensures that all bit relays have cleared to ZERO before being connected to the signals present on the data bus.

At this point in time the bit relays now match the state of the data bus. Bus lines with voltage on them (representing a ONE) cause the corresponding bit relay to close. When LD-A is dropped by Control, K411 opens, restoring voltage to the hold line. This latches the data bus values into the bit relays. K411 opening also restores the ~LD signal causing K412 to close. This removes voltage from the gating relays, which disconnect the register from the bus. In this case, the time delay between reactivation of the hold line and dropping of the gating relays ensures that the values have been latched into the bit relays before the bus is disconnected.

I835–I842 show the value stored in the register. S750 and S751 may be used to manually load or select the register when Debug is on. I843 and I844 indicate when LD-A or SEL-A are activated. C401 is a power bypass capacitor.

Register D is identical to register A. Registers B and C are similar but have an additional output from each bit relay that drives the inputs to the ALU and the Print control section.

Instruction Register

The Instruction Register (*RegisterINST-x*) is similar to registers B and C. Its additional bit outputs connect to the Control section, where it is used to decode the machine instructions. It differs from registers B and C in that it cannot be gated onto the data bus, only loaded from it. Thus there is no select signal input to the load control relays, nor switch for

forcing such a signal. I987 is labeled EN-INST (instead of SEL-INST), to show when the register is being connected (enabled) to the bus for loading.

Register XY

Register XY (*RegisterXY1-x* through *RegisterXY3-x*) is an 8/16-bit register. It can be used on the 8-bit data bus as Register X or Register Y, or on the 16-bit address bus as Register XY. Because it appears as three registers to the rest of the machine, it has three load/select control circuits. K1113 and K1114 control 16-bit operation for XY, K1115 and K1116 control 8-bit operation for X, while K1129 and K1130 control 8-bit operation for Y. Note that the hold voltage source to K1115 and K1129 does not come from V, rather from K1113aNO, the hold line for the 16-bit register. The bit relays, are split into two 8-bit groups: K1101–K1108 for X, and K1117–K1124 for Y. When XY is to be loaded from the 16-bit bus, the register is cleared by dropping the hold voltage for both X and Y. Loads from the 8-bit data bus clear only X or Y, as required.

K1109 and K1111 gate X onto the data bus, while K1125 and K1127 gate Y onto the data bus. All 16 bit relays constituting XY are gated onto the address bus by K1110, K1112, K1126, and K1128.

As for other registers, switches are provided to manually load/select the three registers, along with displays to indicate those conditions. I913–I920 show the contents of X (most significant byte of XY) and I925–I932 show the contents of Y (least significant byte of XY). C1101 and C1102 are power bypass capacitors.

Register M

Register M (*RegisterM1-x* through *RegisterM3-x*) is an 8/16 bit register similar to XY. It can be used on the 8-bit data bus as Register M1 or Register M2, or on the 16-bit bus as Register M. Unlike XY, however, it cannot be loaded from the 16-bit address bus, only selected onto it as a signal source. As a result of this it has no load control relays for the 16-bit M combination of M1 and M2. SEL-M (Select-M) directly controls the 16-bit gating relays to place M on the address bus. Otherwise, it operates the same as Register XY.

Register J

Register J (*RegisterJ1-x* through *RegisterJ3-x*) is yet another variation of an 8/16-bit register. Similar to M, it can be loaded from the 8-bit data bus as J1 and J2, and it can be selected onto, but not loaded from, the 16-bit address bus. Unlike M, J1 and J2 cannot be selected as a signal source onto the 8-bit data bus. The J1 and J2 load control circuits hence have no select inputs. Like M, the 16-bit select input directly drives the address bus gating relays.

Program Counter

The Program Counter (*RegisterPC1-x* through *RegisterPC2-x*) is a 16-bit register, accessible only from the address bus. It is essentially the same as the XY register, minus the 8-bit load control and gating relays.

Increment Register

The Increment Register (*RegisterINC1-x* through *RegisterINC2-x*) is a 16-bit only register with a split personality. It can be selected onto the address bus just like the M register, but it can be loaded only from the output of the Incrementer. Accordingly, it has separate gating relays for the load and select functions.

Relays K480 and K482 are used to put the contents of the bit relays onto the 16-bit address bus, directly controlled by the SEL-INC (Select-Increment) signal. K483 and K484 form a standard load control circuit, but K484aNC drives a different set of gating relays from those used to put contents onto the data bus. In a standard load/select control circuit, the select signal would be tied to the same gating enable path used for loading the register. Here they drive different gating relays because the source and destinations are not the same. The I15–I0 signals used to load the register come from the Incrementer instead of the address bus.

K479, K481, K493, and K495 gate the output of the Incrementer onto the bit relay coils for register loading. K480, K482, K494, and K496 gate a separate output from the bit relays onto the address bus during a select operation.

Condition Code Register

The Condition Code register (*ArithmeticLogicUnit11-x*) is functionally part of the ALU, but its operation is the same as any other register. K563 and K564 are the standard load control relays, but the inputs to gating relay K562 come from the ALU instead of a bus.

The register stores three bits, the results of the last ALU operation: Sign (K559), Carry (K560), and Zero (K561). The outputs are wired directly to the Control section, so there is no select circuit input.

Arithmetic Logic Unit

The Arithmetic Logic Unit (ALU) takes its inputs from the B and C registers and can perform the following functions under program control:

ADD B+C

INC B+1

AND logical AND of B with C

OR logical OR of B with C

XOR logical exclusive OR of B with C

NOT bitwise complement of B

SHL B circularly shifted left by one bit

The C register is disconnected from the adder for the INC (increment) function, and is ignored for NOT and SHL (shift left) functions, which operate only on the contents of the B register.

Function Decoder

Op-codes for ALU operations contain a 3-bit function code (F2–F0) that selects the specific ALU function for that op-code. Drawing *ArithmeticLogicUnit1-x* shows the 3-to-8 decoder that selects which function gets gated onto the data bus.

Relays K501–K503 receive the 3-bit function code from the Control section and turn it into a 1-of-8 selection. The ALU function codes are as follows:

000 Unused (if selected, will place a 0 on the output bus)

001 ADD

010 INC

011 AND

100 OR

101 XOR

110 NOT

111 SHL

S780–S782 may be used to force the ALU function code signals for debugging purposes, and I1017–I1019 indicate the state of these signals. Likewise, S789–S783 can be used to make an ALU function selection, but only when Debug voltage is on. I1020–I1027 indicate the function selected. C501 is a power bypass capacitor.

Adder Control and Enable

ArithmeticLogicUnit2-x shows circuits that control the Register C and carry inputs to the adder, along with the relays that gate the ADD/INC output onto the ALU output bus.

For ADD operations, the contents of the C register (C7–C0) pass through the normally-closed contacts of K504 and K505, which are de-energized, and are fed to the C input of the adder as signals C7'–C0'. Relay K506, also open, places voltage on the ~Cin line and leaves Cin open, corresponding to a ZERO carry, which is fed into the least significant bit of the adder. Relays K507 and K508 gate the output of the adder (O7–O0) onto the ALU output bus (X7-X0) under control of the ADD signal through K506dNC.

For an INC (increment) operation, the INC signal activates K504–K506. K504 and K505 open the C register inputs to the adder, making them zero, and K506 puts voltage on Cin and removes it from ~Cin, passing a ONE carry input to the LSB of the adder. This provides the necessary ONE for the increment operation. K507 and K508 gate the output of the adder onto the ALU output bus, controlled by the voltage applied to K506dF.

Switches S798–S805 may be used to force a C input when Debug is on, and I1036–I1043 show the register C input value. I1044–I1051 display the ADD/INC output. Note that this will be the ADD output unless an INC operation is being performed.

Zero Detection

The ALU output bus (X7–X0) is continuously monitored by the zero detect circuit (*ArithmeticLogicUnit3-x*), which outputs a ONE when all bits are zero. Relays K509–K516 monitor one bit each and if every bit is zero a voltage path is established from K509aNC

through to K516aNC. This Z signal is presented to the Condition Code register where it is latched at the end of each ALU operation, indicating that the ALU result was zero.

Adder

The adder circuit (*ArithmeticLogicUnit4-x*) was designed by Konrad Zuse for the German relay computer Z3, completed in 1941. It is an elegant design that requires only two relays per bit (one for each input), and has no carry ripple delay unlike other adders. When the two inputs are presented to the adder, the sum and the state of the output carry are instantly available. It accomplishes this by passing both the carry (Cin) and its inverse (~Cin) through all stages of the adder. In essence, all possible outputs are prewired and it only takes the inputs to select the correct output.

The upper bank of relays (odd numbers K517–K531) accept the C input (from Register C for an ADD, or zero for an INC) to the adder and the lower bank (even numbers K518–K532) accept the B input (from Register B). The carry input to the least significant bit is provided by the adder control circuit on Cin, along with its inverse on ~Cin. As previously described, this will be a ZERO (Cin low, ~Cin high) for an ADD, and a ONE (Cin high, ~Cin low) for an INC.

The adder output (B+C for an ADD, B+1 for an INC), O7-O0, is passed to the Adder Control and Enable circuit for gating onto the output bus. The carry out of the most significant bit, if any, is present on K518dNC. This CY signal is passed to the Condition Code register where it is latched at the completion of each ALU operation.

Note

The carry output resulting from B+C is latched by the Condition Code register at the end of each ALU operation, even if the selected function is not ADD or INC. Thus branch selection based on the state of the carry bit is only valid following an ADD or INC operation.

S790–S797 may be used to force the B input when Debug is on. I1028–I1035 indicate the B input to the ALU. C502 is a power bypass capacitor.

Logic Unit

The logic unit (*ArithmeticLogicUnit5-x*) calculates the values for the ALU logic functions. AND, OR, and XOR are logical combinations of the bits coming from the B and C registers, whose outputs are as defined by the following truth tables:

B · C	**C**	
B	0	1
0	0	0
1	0	1

AND

B + C	C	
B	0	1
0	0	1
1	1	1

OR

B ⊕ C	C	
B	0	1
0	0	1
1	1	0

XOR

The NOT function inverts the bits of the B register:

B	~B
0	1
1	0

NOT

The SHL function shifts the contents of the B register circularly left (sometimes called a left rotate in other computers) by one bit:

$b_7b_6b_5b_4b_3b_2b_1b_0$ becomes $b_6b_5b_4b_3b_2b_1b_0b_7$

The logic unit consists of eight 1-bit logic circuits, containing two relays each. Looking at bit 7, the B input drives the coil of K517 and the C input drives K518. K517aNO is connected to K518aNO such that both relays need to be closed to produce the AND7 output. K517bNO and K518bNO are arranged so that closing either or both of them places voltage on the OR7 output. Contact set K517c is connected to K518c with a crossover such that the relays have to be in different states (one closed, one open) to produce the XOR7 signal. Finally, K517dNC inverts bit 7's input from the B register by producing a high when the relay is open and a low when it is closed. The SHL7 signal at K517dNO is just a copy of the B input; the left shift is done by wiring at the output selector for SHL.

The remaining seven bits are wired identically to bit 7. Note that all logic functions are computed continuously based on the contents of the B and C registers. When one of these outputs is desired, it is gated onto the data bus by the decoded ALU function code.

Logic Display and Enable

Outputs produced by the logic unit are fed to individual display and enable circuits for each function (*ArithmeticLogicUnit6-x* through *ArithmeticLogicUnit10-x).* All are identical, with one exception.

Inspecting the AND circuit (*ArithmeticLogicUnit6-x*), we see indicators I1052–I1059 that display the results of the AND operation, and relays K549 and K550, which gate the AND output from the logic unit onto the ALU output bus (X7–X0) when the ALU function code selects the AND function.

Display and enable circuits for OR, XOR, and NOT are wired identically to the one for AND. The circuit for SHL is the same except that the bits from the logic unit are shifted down one compared to the outputs, to produce the left shift. Thus output bit X7 is fed by SHL6 from the logic unit, X6 is fed by SHL5, and so on.

Condition Code Register

The Condition Code register (*ArithmeticLogicUnit11-x*) latches the Sign, Carry, and Zero states of ALU operations for use by the conditional branching circuitry in the Control section. It was discussed previously in the Register section of this document.

Result Bus and Display

All ALU outputs (*ArithmeticLogicUnit12-x*) from the adder and the logic unit are gated onto the ALU output bus, X7–X0, which is connected to the ALU output display indicators I988–I995, and to the data bus. X7 is picked off and fed to the Condition Code register as the Sign bit.

Note

The ALU result display was included to allow debugging of the ALU with the rest of the machine disconnected from it. When the data bus is connected, this display shows what is on the data bus and not the ALU output, unless an ALU function has been requested. Control holds the ALU function code at 000 during other than ALU operations. This removes all ALU gating to the data bus, so there is no conflict with other data bus activities.

Incrementer

The incrementer (*Incrementer1-x* and *Incrementer2-x*) continuously adds 1 to the value on the 16-bit address bus, and is used to update the Program Counter with the next sequential address.

It consists of 16 half adders (similar to one bank of the ALU's adder), with a permanently wired ONE (Cin high, ~Cin low) fed into the carry input of the least significant bit. Carry out of the most significant bit, if any, is ignored.

K1201–K1216 receive the 16 bits of input from the address bus (A15–A0) on their coils. K1216aNO is the Cin input, tied to voltage, and K1216bNO is the ~Cin input, left open. Outputs (I15-I0) are fed to the load gating relays on the Increment register, as previously discussed. I1001–I1016 display the incrementer's output. C1201 is a power bypass capacitor.

Instruction Execution

Before delving into the complexities of the Control section, it will be informative to examine the pulse sequences needed to execute the various machine instructions. Recall that the master timing chart (*MasterTiming1-x*) showed the entire set of 19 control pulses and gave some indication of their use. Let us now inspect the control sequences in more detail.

Fetch and Increment

Starting with *InstructionTiming1-x*, we see a section labeled Fetch/Increment. Execution of all instructions starts with this first sequence. Pulse P-A is sent to the select line of the Program Counter; this places the current PC address on the 16-bit address bus. At the same time, P-A is also sent to request a Memory Read to fetch the instruction op-code from memory. One clock tick later (T2, referring to the clock stream at the top of the drawing), the instruction is on the data bus, and its address is on the address bus. P-B is then used to load the instruction into the Instruction Register. At the same time, P-B is also used to load address+1 from the Incrementer into the Increment Register. Note that P-A extends one clock tick longer than P-B. This ensures that the op-code on the data bus and the address on the address bus are stable until latched into a register.

T4 is an idle period allowed to prevent adjacent uses of the address and data bus from interfering with each other (bus activity is shown at the bottom of the drawing). At T5, P-C is sent to select the Increment register onto the address bus. P-D loads the incremented address back into the Program Counter. Again, P-C extends one clock tick longer than P-D to make sure the value on the address bus is stable until it is latched on the falling edge of P-D.

At this point the op-code that was loaded into the Instruction Register in T2 has been decoded by Control, thus determining how many clock periods it needs to execute. If it requires fewer than the maximum of 24, a sequencer abort relay gets set by P-D. From this point on, the pulse sequence depends on the op-code actually loaded.

Each time the sequencer gets reset to start a new instruction, this same Fetch and Increment sequence is performed to load the next instruction and increment the Program Counter.

Move Instructions (8-bit)

Data bus move instructions (MOV8) transfer data from one 8-bit register to another, using the 8-bit data bus. These 8-bit moves can be overlapped with the Program Counter update taking place on the 16-bit address bus. Pulse P-C is used to select the source register onto the data bus, and P-D is used to load that value into the destination register. The registers are determined by 2-bit each source and destination fields in the instruction op-code. These instructions take only 8 clock periods. Note that 16-bit moves (see separate subsection, below) cannot be overlapped with the PC address update because both require use of the address bus.

Print Instruction

The Print instruction was added toward the end of the RC-3 design cycle when an appropriate output device (a Robot Printer) became available. It emits a single pulse to the

Print section using P-D at T5. This instruction requires 8 clock periods and can be overlapped with the PC update on the address bus.

Note

There was no Print instruction in the HPRC. All output for HPRC was from the front-panel indicators.

ALU Instructions

ALU instructions can also be executed in 8 clock periods, overlapped with the PC update on the 16-bit address bus. Pulse P-E sends the function code (determined by a 3-bit field in the op-code) to the ALU. This selects which ALU function gets gated onto the data bus. P-D loads that output from the data bus into either the A or D register, as selected by a 1-bit field in the op-code. P-D is also sent to the Condition Code register to latch the Sign, Carry, and Zero state of the ALU operation.

IncXY Instruction

The XY register is sometimes used as an index register. To support this, the IncXY instruction has been provided to add 1 to the contents of XY. This instruction takes 14 clock periods. At the end of T6 the address bus has been released by the Increment register. T7 is an idle period to separate address bus usages. At T8, pulse P-F selects the XY register onto the 16-bit address bus. That value has 1 added to it by the Incrementer and the updated value is latched into the Increment register by P-G. T10 provides the required idle period on the bus, then P-H selects the Increment register onto the address bus and P-I loads it back into Register XY.

Note

The Condition Code register is not set as a result of INCXY, so no conditional branching can be determined by the result of this operation.

SetAB Instruction

Continuing with *InstructionTiming2-x*, the SetAB instruction loads either the A or B register with a sign-extended 5-bit immediate value contained in the instruction op-code. This instruction can be overlapped with the Program Counter increment update, so can be executed in 8 clock periods. Pulse P-E activates the INST-IMMED-TO-BUS signal, which extends the sign of the most significant bit of the 5-bit field to 8 bits, and gates it onto the data bus. P-D loads that value into either the A or B register as selected by a 1-bit field in the instruction op-code.

Move Instructions (16 bit)

Sixteen-bit moves (MOV16) transfer data across the address bus. No overlap with PC increment update is possible, so these instructions require 10 clock periods. Once the address bus is released in T6, and after the T7 idle period, pulse P-F selects the desired 16-bit source (M, XY, J, or the address entry switches) onto the address bus, as determined by a 2-bit field in the instruction op-code. P-G loads that value into the destination register (XY or PC), as determined by a 1-bit field in the op-code.

Miscellaneous Instructions

Miscellaneous-class instructions include Halt, HLTRL (Halt and Reload), and LDSW (load switches). LDSW transfers the contents of the 8 data entry switches to either the A or D register, using the 8-bit data bus. HLTRL, in addition to the Halt, transfers the contents of the 16 address entry switches to the Program Counter, using the 16-bit address bus. Pulse P-F selects the source onto the appropriate bus, and P-G loads it into the destination register. These instructions take 12 clock periods.

Load Instructions

The Load instructions transfer the 8-bit contents of the memory location addressed by the M register to any one of Register A–D. At T8, pulse P-J selects the M register onto the 16-bit address bus, and sends a read request to memory. Once the byte becomes available, P-K loads it into the destination register, selected by a 2-bit field in the instruction op-code. Load instructions require 12 clock periods.

Store Instructions

The Store instructions transfer the 8-bit contents of any one of Register A–D to the memory location addressed by Register M. At T8, pulse P-J selects the M register onto the 16-bit address bus and issues a Bus-to-Memory request, which puts the 8-bit data bus onto the internal (5V) memory bus. At the same time, P-J selects the desired source register, determined by a 2-bit field in the op-code, onto the data bus. P-K then issues the memory write request. Store instructions take 12 clock periods.

GoTo Instructions

GoTo-class instructions, including SetM, Call, Jump, and the conditional branch variants (BNEG, BC, BZ, BNZ) require 24 clock periods. Their timing is shown on *InstructionTiming3-x*. All GoTo-class instructions are three bytes long: an 8-bit op-code, followed by a 16-bit address. The op-code is read normally, using the Fetch and Increment sequence previously described, then the two bytes of address are read one byte at a time, and transferred to the required destination register (M1/M2 for the SetM instruction or J1/J2 for GoTo, Call, Jump, and conditional branch instructions). After each read a Program Counter increment sequence is required to point to the next byte to be read.

Note

The SetM instruction loads a 16-bit value from memory into the M register (load immediate, 16 bits). It really has nothing to do with the branching operations provided by GoTo, but because it uses the same pulse sequence it was incorporated into the GoTo operations to save relays. This is as it was in the HPRC design.

The chart picks up at the idle point T7, following completion of the Fetch and Increment sequence. In T8, pulse P-J selects the PC onto the address bus, and simultaneously issues a memory read request. At T9, P-K loads the most significant byte of the target address from the 8-bit data bus (result from memory read) into M1 or J1, as determined by a 1-bit destination field (M or J) in the instruction op-code. At the same time, P-K loads the

Increment register with the current address + 1 (provided by the Incrementer). T11 is an idle period between uses of the data bus.

Next, at T12, P-L selects the Increment register onto the 16-bit address bus, and P-M loads the incremented value into the Program Counter, making it point to the next byte to be read from memory. At T15, P-N places the PC contents on the address bus and issues a memory read. P-O loads the value read (least significant byte of the target address) into either M2 or J2, as required, and reloads the Increment register with the current address + 1. At T19, P-Q selects the Increment register onto the address bus and P-R loads the incremented value back into the PC.

If the "copy PC to XY" bit is set in the op-code (for a Call instruction), P-R also loads the value currently on the address bus (next location to read) into the XY register as a return address.

Note

RC-3 has no stack, so only one level of subroutine call is directly supported, using the XY register to store the return address. Deeper subroutine nesting requires the programmer to save and restore the contents of Register XY (its own return address) within the code of each subroutine that calls another one.

Finally, at T22, the J (jump) register is selected onto the address bus. For the Call and Jump instructions, P-T always loads this address into the Program Counter, effecting the jump. For the conditional branch instructions, the LD-PC pulse is emitted only if the condition is satisfied (by testing the contents of the Condition Code register).

Control

The most complex logic in RC-3 is in the Control section. Control receives the 8 bits from the Instruction Register (IR7–IR0), decodes the instruction op-code contained therein, and steers the appropriate sequencer pulses to the right places to execute each instruction. It is constructed with sequential combinatorial logic. Reference to the instruction timing diagrams (described in the last section) may be helpful in determining why each subsection of Control does what it does.

Instruction Class Decoding

Relays K301–K305 (*Control1-x*) determine the basic instruction class using up to the first five bits of the op-code (IR7–IR3). These instruction class signals activate specific subsections of Control to further decode each class of instruction. Instruction classes are displayed by I770–I778. C301 is a power bypass capacitor.

K301–K302 form a 2-to-4 decoder for bits IR7–IR6. Three of its outputs translate directly into instruction classes GOTO, SETAB, and MOV8, each of which uses the remaining bits (IR5-IR0) to provide specific details (described later for each instruction class) for the instruction's execution. The fourth output (corresponding to an op-code of 10xxxxxx) is further decoded.

K303–K304 form another 2-to-4 decoder, which is driven by K302aNC. Two of the four outputs select the INCXY and ALU instruction classes, where again the remaining bits of the

op-code provide specifics for the instruction's execution. The remaining two outputs along with K305 form a 2-to-4 decoder to select among the remaining instruction classes: STORE, LOAD, MISC, and MOV16.

Instruction Fetch and Increment

Drawing *Control2-x* shows the relays that control the Fetch and Increment sequence. Relays K306–K309 are driven by pulses P-A through P-D respectively. Each relay directs a copy of its driving pulse to the appropriate locations to sequence the instruction fetch operation and the PC increment operation. Note that the relays prevent signals back feeding from any of the destinations to any of the sources. The CH-ABT (choose abort) signal gets gated subsequently to a sequencer abort relay if the decoded instruction requires fewer than 24 clock periods.

Load and Store

Load and Store sequencing is shown on *Control3-x*. Relay K345 disables Load and Store if IR2 is set.

Note

IR2 was not decoded for Load and Store in the HPRC design, making a set of duplicate op-codes for both. All duplicate op-codes in RC-3 were forced to be no-ops.

Relays K310 and K311 get energized for Store operations. K310 routes P-J to memory control (BUS-TO-MEM) to gate the 8-bit data bus onto the internal (5V) memory bus. K311 sends P-J to select the M register onto the 16-bit address bus. It also sends P-J to K314aF. The left half of K314 and K315 form a 2-to-4 decoder, which routes P-J to select one of Registers A–D onto the 8-bit data bus, as determined by IR1–IR0. K311 also sends P-K to command the memory write operation (MEM-WRITE). CH-ABT is switched by K311 to the ABORT12 line, setting a sequencer abort relay to shorten the instruction to 12 clock periods.

Relay K312 is set by the Load signal. K312aNO then routes P-J to K313, which selects the M register onto the address bus, and issues a memory read request (MEM-READ). K312 sends P-K to the right half of K314 and K315, which form a 2-to-4 decoder based on IR1–IR0 to route P-K to load the appropriate register (A–D). CH-ABT is also gated through K312 to the ABORT12 line, the same as for Store operations.

SetAB

The SetAB instruction closes relay K316 (*Control4-x*). Pulse P-E is routed through K316aNO to K317 and K318. Recall that this instruction takes a 5-bit value from the op-code, sign extends it, and loads the result into either Register A or Register B. K317 gates IR3–IR0 onto the 8-bit data bus. K318 gates IR4 to bits D7–D4 of the data bus, thus performing the sign extension. K316 sends P-D to K303, which determines which register gets loaded, based on IR5. K316 also sends CH-ABT down the ABORT8 line, setting a sequencer abort relay to shorten the instruction sequence to 8 clock periods. C302 is a power bypass capacitor.

IncXY

Continuing with *Control4-x*, the middle section of the drawing shows control sequencing for the IncXY and Print instructions. Relays K345–K347 eliminate a number of op-codes that were duplicates of IncXY.

Note

In the HPRC design, bits IR3–IR0 were not decoded, resulting in 16 numeric op-codes that produced an IncXY operation. In RC-3, the 15 duplicate op-codes were forced to be no-ops. One op-code from this set of 15 no-ops was selected for the Print operation. Although it has nothing to do with incrementing the XY register, it could be implemented in the IncXY class without adding any relays. There was no Print operation in the HPRC design.

K348 decodes IR0 to distinguish between the IncXY (10110000) and Print (10110001) op-codes. If bit 0 of the op-code is 0, K348dNC directs the IncXY signal to the coil of K319. K319 then directs pulses P-F and P-G to select XY onto the address bus and load its increment into the increment register, then P-H and P-I to select the increment register onto the address bus and load the incremented value back into Register XY. D375 and D381 prevent signals back feeding into P-F and P-G during this operation.

If bit 0 of the op-code is 1, K348 disconnects the IncXY signal from K319, and gates P-D through to K320bF. K320 is always driven by the IncXY signal, resulting in P-D being sent to the Print section. K320 couples CH-ABT to K348aF, where it gets sent to either ABORT8 or ABORT14 depending on the state of IR0. The Print instruction takes only 8 clock periods, whereas the IncXY instruction takes 14 clock periods.

ALU

Sequencing for the ALU instructions is handled by the last section of *Control4-x*. Relay K321 is activated by the ALU class signal. This routes pulse P-E to close K322, sending the three bits of ALU function code (from IR2–IR0) on to the ALU. K321 also sends P-D to load the condition register and to load the destination register, selected by K305 based on bit IR3. ALU instructions require only 8 clock periods, so K321 gates CH-ABT to the ABORT8 line.

Move (8-bit)

The MOV8 instruction is sequenced by the left half of *Control5-x*. Relays K324–K326 form a 3-to-8 decoder that selects the destination register based on IR5–IR3. The left side of relays K327–K329 form a 3-to-8 decoder that selects the source register based on IR2–IR0. Relay K323 is closed by the MOV8 instruction class signal. It then gates P-D to the destination selection tree, and P-C to the source selection tree. These instructions take only 8 clock periods, so K323 gates CH-ABT to the ABORT8 line. D373 and D374 prevent signal back feeding from LD-A and SEL-A.

Move (16-bit)

The MOV16 instruction is controlled by the middle portion of *Control5-x*. K330 is closed by the MOV16 instruction class signal, sending P-G to the destination register, determined by IR2 using K327b. P-F gets gated through a 2-to-4 decoder, formed by K328c and the left

half of K344, to one of the four source locations. These instructions take 10 clock periods, so K330 also gates CH-ABT to the ABORT10 line. D372 prevents signal back feed from LD-PC.

Note

In the HPRC design, bit IR3 was not decoded for MOV16 instructions, resulting in 7 duplicate op-codes. A new instruction class was created in the RC-3 design, called Miscellaneous, to use these duplicate op-codes for other purposes.

Miscellaneous

The miscellaneous instructions (Halt, HLTRL, and LDSW) are shown on the right side of *Control5-x*. The Miscellaneous instruction class signal is gated through K327c to K343 if IR2 is a 1. Pulse P-F selects a source (data entry switches or address entry switches) for the LDSW (load switches) and HLTRL (halt and reload PC) instructions. If K328 is open (IR1=0), the data entry switches are selected onto the data bus. If K328 is closed (IR1=1), P-F is sent on to K344 where it is used to select the address entry switches onto the address bus if IR0=1. If IR0=0, no source is selected. K343 also sends P-G to K314, where it is steered based on IR1 to either K344 for selection of a destination register (for LDSW), or to the HALT line and K348 (for Halt or HLTRL). If IR0=1 (K348 closed), P-G gets sent to load the PC, in addition to triggering the halt (for HLTRL). If K348 is open (IR0=0), the PC is not loaded, and only the halt signal is driven by P-G. D349 and D350 prevent signal back feed on the LD-PC and HALT lines.

GoTo and Variants

Drawing *Control6-x* diagrams control circuitry for the complex GoTo operation, along with the SetM, Call, and conditional branch (BNEG, BZ, BNZ, and BC) variants of it. Recall that all GoTo class instructions are three bytes long, and require two load/increment cycles to fetch the two byte address following the op-code. These instructions require 24 clock periods, so there is no abort selection.

K331–K332, and K334 are all activated by the GoTo instruction class signal. K331 sequences the two memory reads required, by sending P-J and P-N to select the PC onto the address bus, and by sending the same pulses to request a memory read. D372 prevents signal back feed from SEL-PC.

The first byte read is loaded into the destination register (M1 for SetM, or J1 for all others, as selected by IR5 using K303) by pulse P-K, which gets repeated by the closure of K336. At the same time the byte from memory is being loaded via the 8-bit data bus, the increment register is being loaded with the next address using the 16-bit data bus, driven by the LD-INC signal produced by K336bNO.

The second byte read is then loaded into its destination register (M2 for SetM, or J2 for all others) in a similar fashion by K335, which repeats pulse P-O to load the data byte and load the increment register. Note that K335 and K336 are required to prevent signal back feed from the driven LD-x lines.

The Program Counter is updated with the incremented address by selecting the increment register onto the address bus using P-L for the first update, and P-Q for the second update,

both via K331, then using P-M and P-R repeated by K333, to load the PC. D370 and D371 prevent signal back feed.

If this is a Call instruction (IR0=1, the x control bit), the XY register is also loaded with the address following the second increment, by gating IR0 through K333b under control of pulse P-R. This saves the next address to be read in XY as a return point for the subroutine call.

The final step in processing a GoTo class instruction is to determine whether or not to jump to a new location. In all cases, P-S selects the J register onto the 16-bit address bus through K334a, then a decision is made whether to load the PC or not. K304–K305, K327, and K314 are each controlled by condition bits in the op-code (IR4=s, IR3=c, IR2=z, and IR1=n, respectively). If a particular control bit is set, the corresponding value from the Condition Code register is gated through contact set d of the relay. K304 gates the Sign state, K305 the Carry, K327 Zero, and K314 Not Zero. These signals (all protected from back feeding to each other by D366–D369) are combined in a wired-OR connection feeding K337d. If any of the gated signals is a 1 (op-code control bit set AND its corresponding condition code set), then P-T loads the PC via K337.

Refer to the instruction set document (*RC-3InstructionSet-x*) for details on how to interpret various combinations of the GoTo op-code's control bits. Just note here that a SetM instruction will have none of the control bits set, so the jump is never taken, and a Jump (unconditional branch) will have both the z (zero) and n (not zero) control bits set. Since CZ and ~CZ are always complements of each other, this guarantees that the PC gets loaded from the J register and the branch is taken. The conditional branches depend on whether any control bits match their corresponding condition codes.

C303 is a power bypass capacitor.

Control Display and Debugging

I785–I821 (*Control7-x* through *Control9-x*) display the 37 control signals produced by the Control section. Inputs to the Control section (IR7-IR0, and the condition codes) are displayed by I822–I833 (*Control10-x*). Switches S735–S745 may be used to force input signals whenever Debug is on. During normal operation the Condition Code register complements input CZ to produce ~CZ. If the Control section is being debugged with the Condition Code register disconnected, JM801 may be installed to allow both CZ and ~CZ to be produced by S745.

Front Panel Operations

As mentioned in the design overview, RC-3 provides convenient front panel switches for reading and writing memory, and loading addresses into the Program Counter. Each of these front panel operations switches performs actions that would require several switch operations in the HPRC design.

Front Panel Operations Timing

Pulse sequences used to perform the front panel operations are shown on *FPopsTiming1-x* and *FPopsTiming2-x*. Four pulses, called AP-A, AP-B, AP-C, and AP-D (AP for auxiliary pulse),

produced by the Auxiliary Clock (described below), are used to execute the front panel operations.

Starting with *FPopsTiming1-x*, we see that the Load Address operation is performed by sending AP-A to select the address entry switches onto the 16-bit address bus, and using pulse AP-B to load that value into the Program Counter.

Examine reads the memory location pointed to by the Program Counter and stores it in Register A. To do this, it sends AP-A to select the PC onto the 16-bit address bus, and issues a memory read request. It then uses AP-B to load the byte read from memory into the A register.

Deposit writes the contents of the data entry switches into the memory location pointed to by the Program Counter. Pulse AP-A is used to select the PC onto the 16-bit address bus and send the Bus-to-Memory signal to gate the 8-bit data bus onto the memory's internal (5V) data bus. AP-A is also used to select the data entry switches onto the data bus. Finally, AP-B is used to issue the memory write request.

All three of these operations can be executed using only the two pulses AP-A and AP-B. As we will see shortly, each of them sets the Auxiliary Clock abort relay to shorten the pulse sequence generated to these two pulses.

The Examine Next and Deposit Next sequences are shown on *FPopsTiming2-x*. Each performs its memory read or write, as described above, then increments the Program Counter to its next address. This is convenient for sequentially reading or writing a block of memory.

Examine Next reads a memory location in the same way as Examine, but while the byte from memory is being transferred to the A register on the 8-bit data bus, pulse AP-B is also being used to load the Increment Register with PC+1 from the Incrementer. After these transfers are complete, pulse AP-C selects the Increment Register onto the 16-bit address bus, and AP-D loads it into the PC.

Likewise, Deposit Next writes a memory location in the same way as Deposit, and increments the Program Counter the same way as Examine Next.

Auxiliary Clock

The Auxiliary Clock (*AuxClock-x*) produces the four auxiliary pulses used to perform the five front panel operations. Relays K106–K111 form a 6-stage ring counter with delay at each stage, similar to the master clock. Unlike the master clock, however, there is no feedback from the end back to the first stage, so the aux clock runs once through each time it is triggered. K112 is the aux clock reset relay, and K113 is an abort relay used to shorten the clock sequence. C105–C112 provide timing delays for each relay, and C114 is a power bypass capacitor.

The auxiliary pulses are generated directly directly from the clock relays, so no separate sequencer is required. The pulse sequences are shown on *AuxClockTiming-x*.

Each time a front panel operation is requested, the AUX-START signal is generated. This sets K106, which remains closed for an interval (determined by C105) after the start signal is removed. K106bNO then powers K107. Once K106 opens, voltage becomes available at K107bNO and K108 closes. This process continues down the line: as each relay opens, it sets the second one following it. If K113 is open (no abort), the sequence continues until

K112 gets set, then as C111 depletes, K112 opens and the sequence ends. K112 is used to reset the aux clock abort relay (K113) and relays in the Front Panel Operations circuits via ~AUX-RESET.

Operations requiring only pulses AP-A and AP-B result in an AUX-ABORT signal being sent to set K113, which locks up on the voltage it receives from K112dNO through K113bNO. In this case, when K107 drops out instead of setting K109, K112 gets set through K113aNO. K112 closes, removing the lockup voltage to K113, which then drops out, resetting the abort, and the clock stops.

I707–I713 indicate the state of the aux clock relays. I714 shows when the abort relay is set, and I715–I718 show when the auxiliary pulses are being produced. Switch S705 can be used to manually start the aux clock when Debug is on, and S706 can be used to set the abort relay, likewise only when Debug is on. Switches S707–S710 can be used to force the auxiliary pulses when in Debug mode.

Front Panel Operations Sequencing

Control circuitry for the front panel operations is shown on *FrontPanelOpns-x*. Switches S730–S732 actuate the front panel operations. S730 is an OFF-ON switch, spring loaded in the off direction. Switches S731 and S732 are both ON-OFF-ON switches, spring loaded to their center off position.

Each of the switches generates an AUX-START signal, to start the auxiliary clock. These are all isolated from one another by diodes D356–D360. Recall that each of the operations Load Address, Examine, and Deposit, can be executed using only two pulses. Each of those operations requires generation of an AUX-ABORT signal to set the aux clock abort relay. Diodes D351, D352, and D354 prevent back feeding among the three switch inputs to the abort line.

K338 controls the Load Address operation. When S730 is closed momentarily, K338 closes and locks closed on the ~AUX-RESET voltage from the auxiliary clock. The aux clock produces AP-A and AP-B, which are routed by K338 to select the address switches onto the 16-bit address bus, and to load that value into the Program Counter. The closing of S730 also sends an AUX-ABORT signal through D351 (to shorten the aux clock sequence to only two pulses) and an AUX-START signal through D356 to start the aux clock running. When the auxiliary clock reset relay K112 closes, this removes voltage from ~AUX-RESET and K338 is released, completing the operation.

K339 and K341 control the Examine and Examine Next operations. If S731 is moved to its up position for an Examine, K339 is powered through D353, and it locks up on ~AUX-RESET. An abort signal is sent through D352 and an aux clock start signal is sent through D357. K339bNO activates K341, which routes AP-A to select the PC onto the address bus (through D379 for back feed prevention) and to issue a memory read request through D380 (again for back feed prevention). K341 also sends AP-B to load the A register, and to load the PC+1 incremented address into the increment register. Note that K339 has also connected AP-C to select the increment register onto the 16-bit address bus (through D377) and AP-D to load the PC (through D376), but because the aux abort relay gets set, the aux clock does not generate AP-C and AP-D, so the PC increment does not take place; the value loaded into the increment register is not used. When the aux clock removes power from ~AUX-RESET, K339 is released, dropping K341, and the operation is complete.

If S731 is moved to its down position for an Examine Next, operation is similar to that for Examine. K339 gets set through D364 but no abort signal is produced. This time the aux clock produces all four auxiliary pulses and the incremented address is transferred from the increment register back into the Program Counter.

K340 and K342 control Deposit and Deposit Next operations. These are handled in a similar fashion to Examine and Examine Next. Deposit closes K340 (through D355), which locks up on ~AUX-RESET, and closes K342. AP-A selects the PC onto the 16-bit address bus, gates the relay address bus onto the memory's internal (5V) address bus (the B2M signal), and selects the data entry switches onto the 8-bit data bus. Pulse AP-B loads the increment register with PC+1 and issues the memory write request. For a Deposit, an aux abort signal is produced through D354, and pulses AP-C and AP-D are not generated. For Deposit Next, the abort signal is not produced (K340 is closed through D365), so all four pulses are generated. AP-C selects the increment register onto the 16-bit address bus and AP-D loads the PC with the incremented address. When ~AUX-RESET goes low, K340 is released, dropping K342, and the operation is complete.

I780–I782 indicate when a front panel operation relay is closed. Note that S730–S732 are fed by FP-OPS-PWR, which is on only when the run/stop switch is in the stop position, as shown on *Clock-x*.

Print Driver

The printing output device for RC-3 is called a Robot Printer, designed to sit on top of an IBM Selectric typewriter keyboard. It contains a set of 49 solenoids, arranged in a 7 x 7 matrix, plus a separate solenoid for the shift key. Printing is accomplished by energizing an appropriate solenoid (with or without concurrently actuating the shift solenoid), which strikes a typewriter key thus printing a character.

Energizing a solenoid consists of applying +24V to a pin on one axis of the matrix while applying ground to a pin on the other axis. This activates the solenoid at the selected crosspoint of the matrix. The shift solenoid is controlled separately (by applying ground to its control pin) to determine whether the typewriter shift key is depressed.

Output from the print driver is controlled by a printer code loaded into the B register. Printer codes (*PrinterCodes-x*) contain three fields: two 3-bit fields to make a selection for each axis of the print matrix, plus a 1-bit field to control the shift key. Normally, the printer code table is stored in memory, arranged in ASCII code order (see the Hello World programming example, below). An ASCII character to be printed is used as an index into the printer code table. The retrieved printer code is loaded into Register B and then the Print instruction is executed.

The driving circuit for the Robot Printer is shown on *PrintDriver-x*. Relay K1401, driven by bit B6, grounds K1401dF whenever B6=1, activating the shift solenoid on the Robot Printer, which depresses the typewriter shift key. I1416 indicates when the shift solenoid is energized.

Relays K1405–K1407 form a 3-to-8 decoder controlled by bits B2–B0, which places a ground on one of the 7 pins connected to the ground axis of the print matrix. One of the 8 outputs (corresponding to bits 000) is unused. I1401–I1407 indicate which pin has been selected. Note that since the selected pin is grounded, the 12V LED indicators for this axis have all their anodes connected to +12V.

K1408 is closed by the PRINT output pulse from the Control section. It connects +24V to the 3-to-8 decoder formed by relays K1402–K1404, driven by bits B5–B3. This decoding tree applies the 24V signal to one of the 7 pins connected to the powered axis of the print matrix. Again, one output from the decoder is unused. I1408–I1414 indicate when a pin in this set has been energized. Note that the driving voltage for these pins comes from a 24V source. The indicators are 12V LED devices, so their cathodes are all connected to 12V, resulting in a 12V difference between the anode and cathode for an activated indicator.

C1401 is a bypass capacitor across the 24V supply, which is also connected directly to the Robot Printer to provide power to the shift solenoid.

Power Supplies and Distribution

RC-3 requires three operating voltages: 12V @ 300W for relay coils and indicators, 5V @ 20W for memory logic and displays, and 24V @ 70W for the Robot Printer. Drawing *Power2-x* shows the power supplies and their AC feed. S734 is the power on/off switch. It contains a built-in neon indicator to show when power is on. Fuse F711 protects against excessive line current caused by a power supply input short circuit.

Drawing *Power1-x* shows how the 12V is distributed. The indicated sections of the machine are independently fused by F701–F710, each of which has an indicator (I1330–I1339) to show that the fuse in that position is good. Power from the 12V supply is routed through meter M701 to monitor DC current drawn by the relays.

S733 is a key-operated switch used to enable debugging. As noted in previous descriptions, switches used to force signals during troubleshooting activities are only active when the Debug switch is closed, providing V_d to the front panel. I1340 indicates when Debug mode is active. C701 and C702 are power bypass capacitors.

Physical power distribution is shown on *Interconnections2-x*. Power from the DC fuses, plus DC grounds, are tied to a set of barrier terminal strips. Power is then distributed from the terminal strips to mini-Molex connectors on each relay rail and front panel section. This allows easy disconnects should a section of the machine need to be removed for maintenance. Note that the LED indicators attached to each DC fuse are not shown on this drawing for clarity.

System Interconnections

Drawing *Interconnections1-x* shows how the various sections of the machine are connected together. All cables in the middle section of the drawing are standard 25-pin data cables with DB-25 male connectors on each end. All Pxxx connectors (except P103 and P204) are DB-25 females mounted on printed circuit boards. Cable pinouts are listed in document *CablePinOuts-x*.

The five boards down the center of the drawing (P7xx series) connect to various sections of the front panel. Connectors to the logic of the machine are mounted on relay connector boards attached to the relay rails. Connectors with the same number but different suffixes (A or B) are paralleled on the relay connector boards to facilitate daisy chained connections.

P103/P204 is a single-pin bullet connector that passes the clock signal from the clock rail to the sequencer.

Lines shown in green represent cables carrying the 8-bit data bus and the 16-bit address bus, which daisy chain through all relay rails that need bus access. The 37 signal outputs produced by the Control section are bused to receiving locations via the orange lines. Other, localized, control signal lines are blue in color. The purple line is a bused display connection that carries signals for both memory and instruction display to the front panel. All other cables carrying front panel display and switch signals are drawn in black.

Not shown on this drawing are the connections between memory rails 2 & 3 and the 5V memory logic board. Those connections are made using 34-wire flat cables with Berg connectors. Their pin assignments are shown at the end of *CablePinOuts-x.*

Sample Programs

Programs for the RC-3 can be prepared using the Cross-32 cross assembler running in a Windows environment, configured by an RC-3-specific data table, *GCMRLxx.tbl*. Test code source for the assembler is in *EGCMRLxx.asm*, with its output listing in *EGCMRLxx.lst*. The comment field for each instruction shows the expected compiled value.

Note that some assembler mnemonics differ from the hardware name for instructions. This was done for programmer convenience and to more closely align the mnemonics with current assembly language practice, although in most instances the original hardware name can be used as well. Here are some examples:

LIN (Load Immediate Nibble) = SetAB - loads 5-bits from op-code into register A or B

LI (Load Immediate) = SetM - loads the M register with a 16-bit value

MOVB (Move Byte) = MOV8 - 8-bit move, a substitution made by the RPTXT directive

MOVL (Move Long) = MOV16 - 16-bit move, a substitution made by the RPTXT directive

CLR (Clear) - clears are performed by moving a register to itself

Memory Test

The following program will execute a test on SRAM memory. It starts at a 256-byte block selected by the value in the data entry switches when the program is started, defaulting to starting block 0 if the block number is invalid. It pauses after reading the starting block location to accept a number of blocks to test, also entered using the data entry switches.

If the test is good, the program halts with 803Ah in the Program Counter. Restarting from the halt causes the program to jump back to the beginning. Two patterns are used for the test. If a write/read fails on the AAh pattern, the program stops with 8032h in the PC and the failing address in Register M. Likewise for a write/read failure on the 55h pattern, the program stops with 8036h in the PC and the failing address in Register M.

This program was loaded into the EEPROM, so its starting address was ORG'd to 8000h, the first address in PROM space.

```
;*****************************************************
; Memory test program for the RC-3 relay computer
```

```
                ; Rev 1          15 Aug 11

0000                  CPU     "GCMRL16.TBL"   ;PROCESSOR TABLE
0000                  HOF     "INT8"    ;HEX FORMAT, 8 bits
0000                  TITL    "RC-3 HANDY DEMOS"
0000                  PAGE    0

                ;*************************************************
                ;
                ; Set 256 byte block (00 - 7F) to test in data switches before running
                ; Program halts to allow entry of number of locations to test (01-FF)
                ; Reset data switches with number of locations to test, and press
                ; RESTART

8000                    ORG   8000H

8000 C055AA     START:  LI    55AAH    ; load test pattern into M
8003 A0                 MOV16 XY,M     ; save it in XY
8004 AC                 LDSW  A        ; get memory block to test
8005 08                 MOV8  B,A      ;
8006 8E                 NOT   D        ; invert to check the msb
8007 F0800B             BNEG  BLKSET   ; ok if msb now set
800A 00                 CLR   A        ; default to zero if block not valid
800B 20         BLKSET: MOV8  M1,A     ; set MSB of memory address
800C 2D                 CLR   M2       ; set LSB of memory address
800D AE                 HALT           ; wait for test count

                ; ********************************************************************
                ;
                ; ENTER NUMBER OF LOCATIONS TO TEST
                ; IN DATA SWITCHES WHEN MACHINE HALTS,
                ; THEN RESTART
                ;
                ; ********************************************************************

800E AC                 LDSW  A        ; get the count
800F 08                 MOV8  B,A      ;
8010 8F                 SHL   D        ; ALU inst to set condition codes
8011 E28015             BNZ   CNTSET   ; ok if count not zero
8014 68                 LIN   B,8      ; default to 8 if count not valid
8015 8E         CNTSET: NOT   D        ; invert count
8016 0B                 MOV8  B,D      ;    and
8017 8A                 INC   D        ; convert to negative number

                                       ; initialization now complete

8018 16         LOOP:   MOV8  C,X      ; get first test byte (AAH)
8019 9A                 STORE C        ; write it [M]<-C
801A 91                 LOAD  B        ; read it back into B
801B 85                 XOR   A        ; compare
801C E28031             BNZ   FAILA    ; any difference indicates error
801F 17         GOTO5:  MOV8  C,Y      ; get second test byte (55H)
8020 9A                 STORE C        ; write it [M]<-C
8021 91                 LOAD  B        ; read it back into B
8022 85                 XOR   A        ; compare
8023 E28035             BNZ   FAIL5    ; any difference indicates error
8026 0B         NEXT:   MOV8  B,D      ; fetch the count
8027 8A                 INC   D        ; increment it
8028 E48039             BZ    GOOD     ; finished if zero
802B 0D                 MOV8  B,M2     ; fetch current memory location
```

```
802C 82                 INC    A         ; increment it
802D 28                 MOV8   M2,A      ; put it back in M for next iteration
802E E68018             JUMP   LOOP      ; go around again

8031 AE          FAILA: HALT             ; failed test pattern A
                                         ; note address of failure, then RESTART
8032 E6801F             JUMP   GOTO5     ; continue testing with pattern 5
8035 AE          FAIL5: HALT             ; failed test pattern 5
                                         ; note address of failure, then RESTART
8036 E68026             JUMP   NEXT      ; continue to increment steps
8039 AE          GOOD:  HALT             ; all locations test OK if stopped here
                                         ; reset block to test in data switches
                                         ; and RESTART to continue testing
803A E68000             JUMP   START
```

Simple Counter

This program counts the A and C registers up to the value entered into the data entry switches when the program starts. If the data entry switches are set to zero, the count defaults to 8. Restarting from the halt at the end of the program jumps to the beginning and runs it again. This demo program was also installed in EEPROM, so was ORG'd to starting location 8100h.

```
                ;******************************************************
                ; Simple counter program for the RC-3 relay computer
                ; Rev 1           15 Aug 11
                ;******************************************************
                ;
8100                    ORG    8100H

                ; Set number to count to in data switches before running
                ; Defaults to 8 if zero

8100 AC         START2: LDSW   A         ; get the desired max count
8101 08                 MOV8   B,A       ;
8102 8F                 SHL    D         ; ALU inst to set condition codes
8103 E28107             BNZ    CNTST2    ; ok if count not zero
8106 68                 LIN    B,8       ; default to 8 if count not valid
8107 8E         CNTST2: NOT    D         ; invert count
8108 0B                 MOV8   B,D       ;    and
8109 8A                 INC    D         ; convert to negative number
810A 00                 CLR    A         ;
810B 12                 CLR    C         ;

                ; initialization now complete

810C 08         LOOP2:  MOV8   B,A       ; get displayed count
810D 82                 INC    A         ; increment and display in A
810E 10                 MOV8   C,A       ; also display in C
810F 0B                 MOV8   B,D       ; get loop counter
8110 8A                 INC    D         ; increment it
8111 E48117             BZ     DONE      ; finished if zero
8114 E6810C             JUMP   LOOP2     ; go again
8117 AE         DONE:   HALT             ; put new loop count in data switches
                                         ; and RESTART to continue
8118 E68100             JUMP   START2    ;
```

Hello World

No computer would be complete without a "Hello world" program. This demo shows the complexities involved in using the Robot Printer for printed output, as well as subroutine calls. The program was installed in EEPROM starting at address 8400h.

```
                ;*****************************************************
                ; Hello world demo for the RC-3 relay computer
                ; Rev 1          18 Sep 11
                ;*****************************************************

8400                    ORG    8400H

8400 C00019       HELLO: SETM   LEN      ; length of msg
8403 0D                 MOV8   B,M2     ;
8404 E7E107             CALL   NEGATE   ; loop count in D
8407 00                 CLR    A        ; zero the offset
8408 C08A00       HCONT: SETM   MESG     ; get base of msg array
840B 28                 MOV8   M2,A     ; insert offset
840C 91                 LOAD   B        ; get a char
840D E7E100             CALL   CHROUT   ; print it
8410 08                 MOV8   B,A      ; get offset
8411 82                 INC    A        ; increment it
8412 0B                 MOV8   B,D      ; get loop counter
8413 8A                 INC    D        ; increment it
8414 E28408             BNZ    HCONT    ; finished?
8417 AE                 HALT            ; yes
8418 E68400             JUMP   HELLO    ; release halt to repeat

8A00                    ORG    8A00H

8A00 48656C6C6F20 MESG: DFB    "Hello "
8A06 776F726C6421       DFB    "world!"
8A0C 0D                 DFB    0DH
8A0D 52432D3320         DFB    "RC-3 "
8A12 686572652E         DFB    "here."
8A17 0D                 DFB    0DH
8A18 0D                 DFB    0DH
0019 =            LEN:  EQU    $-MESG


                ;*****************************************************
                ; Print subroutine for the RC-3 relay computer
                ; Rev 1          18 Sep 11

                ;   Input:    ASCII char in B
                ;   Output:   print code in B
                ;   Output:   print pulse
                ;   Modified: register M
                ;*****************************************************

E100                    ORG    0E100H

E100 C0E000     CHROUT: LI     CHRTBL   ; load base of table
E103 29                 MOV8   M2,B     ; load offset
E104 91                 LOAD   B        ; look up print code
E105 B1                 PRINT           ; send to printer
E106 A5                 RETURN
```

```
                ;*****************************************************
                ; Negate subroutine for the RC-3 relay computer
                ; Rev 1          18 Sep 11

                ;   Input:  0-255 in B
                ;   Output: 2's complement of B in D
                ;*****************************************************

E107 8E         NEGATE: NOT    D        ; complement B to D
E108 0B                 MOV8   B,D      ; move it back to B
E109 8A                 INC    D        ; add one
E10A A5                 RETURN


                ;*****************************************************
                ; Robot Printer lookup table for the RC-3 relay computer
                ; Rev 1          18 Sep 11
                ;*****************************************************

E000            CHRTBL: ORG    0E000h

E000 00         t000:   DFB    0
E001 00         t001:   DFB    0
E002 00         t002:   DFB    0
E003 00         t003:   DFB    0
E004 00         t004:   DFB    0
E005 00         t005:   DFB    0
E006 00         t006:   DFB    0
E007 00         t007:   DFB    0
E008 35         t010:   DFB    35h
E009 0A         t011:   DFB    0ah
E00A 3D         t012:   DFB    3dh
E00B 3D         t013:   DFB    3dh
E00C 00         t014:   DFB    0
E00D 00         t015:   DFB    0
E00E 00         t016:   DFB    0
E00F 00         t017:   DFB    0
E010 00         t020:   DFB    0
E011 00         t021:   DFB    0
E012 00         t022:   DFB    0
E013 00         t023:   DFB    0
E014 00         t024:   DFB    0
E015 00         t025:   DFB    0
E016 00         t026:   DFB    0
E017 00         t027:   DFB    0
E018 00         t030:   DFB    0
E019 00         t031:   DFB    0
E01A 00         t032:   DFB    0
E01B 00         t033:   DFB    0
E01C 00         t034:   DFB    0
E01D 00         t035:   DFB    0
E01E 00         t036:   DFB    0
E01F 00         t037:   DFB    0
E020 3F         t040:   DFB    3fh
E021 49         t041:   DFB    49h
E022 76         t042:   DFB    76h
E023 59         t043:   DFB    59h
E024 61         t044:   DFB    61h
E025 69         t045:   DFB    69h
```

```
E026 79          t046:   DFB    79h
E027 36          t047:   DFB    36h
E028 55          t050:   DFB    55h
E029 5D          t051:   DFB    5dh
E02A 4D          t052:   DFB    4dh
E02B 6D          t053:   DFB    6dh
E02C 27          t054:   DFB    27h
E02D 65          t055:   DFB    65h
E02E 2F          t056:   DFB    2fh
E02F 37          t057:   DFB    37h
E030 1D          t060:   DFB    1dh
E031 09          t061:   DFB    09h
E032 11          t062:   DFB    11h
E033 19          t063:   DFB    19h
E034 21          t064:   DFB    21h
E035 29          t065:   DFB    29h
E036 31          t066:   DFB    31h
E037 39          t067:   DFB    39h
E038 0D          t070:   DFB    0dh
E039 15          t071:   DFB    15h
E03A 5F          t072:   DFB    5fh
E03B 15          t073:   DFB    15h
E03C 00          t074:   DFB    0
E03D 2D          t075:   DFB    2dh
E03E 00          t076:   DFB    0
E03F 77          t077:   DFB    77h
E040 51          t100:   DFB    51h
E041 59          t101:   DFB    59h
E042 6C          t102:   DFB    6ch
E043 5C          t103:   DFB    5ch
E044 5B          t104:   DFB    5bh
E045 62          t105:   DFB    62h
E046 63          t106:   DFB    63h
E047 6B          t107:   DFB    6bh
E048 73          t110:   DFB    73h
E049 56          t111:   DFB    56h
E04A 7B          t112:   DFB    7bh
E04B 4F          t113:   DFB    4fh
E04C 57          t114:   DFB    57h
E04D 7C          t115:   DFB    7ch
E04E 74          t116:   DFB    74h
E04F 5E          t117:   DFB    5eh
E050 66          t120:   DFB    66h
E051 52          t121:   DFB    52h
E052 6A          t122:   DFB    6ah
E053 5A          t123:   DFB    5ah
E054 72          t124:   DFB    72h
E055 4E          t125:   DFB    4eh
E056 64          t126:   DFB    64h
E057 5A          t127:   DFB    5ah
E058 54          t130:   DFB    54h
E059 79          t131:   DFB    79h
E05A 4C          t132:   DFB    4ch
E05B 00          t133:   DFB    0
E05C 00          t134:   DFB    0
E05D 00          t135:   DFB    0
E05E 71          t136:   DFB    71h
E05F 25          t137:   DFB    25h
E060 00          t140:   DFB    0
E061 19          t141:   DFB    19h
```

```
E062 2C          t142:   DFB    2ch
E063 1C          t143:   DFB    1ch
E064 1B          t144:   DFB    1bh
E065 22          t145:   DFB    22h
E066 23          t146:   DFB    23h
E067 2B          t147:   DFB    2bh
E068 33          t150:   DFB    33h
E069 16          t151:   DFB    16h
E06A 3B          t152:   DFB    3bh
E06B 0F          t153:   DFB    0fh
E06C 17          t154:   DFB    17h
E06D 3C          t155:   DFB    3ch
E06E 34          t156:   DFB    34h
E06F 1E          t157:   DFB    1eh
E070 26          t160:   DFB    26h
E071 12          t161:   DFB    12h
E072 2A          t162:   DFB    2ah
E073 1A          t163:   DFB    1ah
E074 32          t164:   DFB    32h
E075 0E          t165:   DFB    0eh
E076 24          t166:   DFB    24h
E077 1A          t167:   DFB    1ah
E078 14          t170:   DFB    14h
E079 3A          t171:   DFB    3ah
E07A 0C          t172:   DFB    0ch
E07B 00          t173:   DFB    0
E07C 00          t174:   DFB    0
E07D 00          t175:   DFB    0
E07E 00          t176:   DFB    0
E07F 00          t177:   DFB    0
```