RK05 Tester, Tests Description

Overview

- Drive Addressing commands allow testing of the drive select signals and sets the drive address used to communicate with the drive under test.
- Seek tests allow testing of drive seek functions and verification of signals for: cylinder address, restore, strobe, and acknowledge signals that result from strobing a new cylinder address.
- Read & Write tests provide the ability to verify reading and writing between tester and drive under test. Some commands are useful for scope loop tests.
- Read & Write disk images tests provide ability to transfer data between drive under test and the tester DRAM. There are also commands to transfer between tester DRAM and the microSD card.
- Miscellaneous commands:
    - Read back drive status
    - Read & write SPI registers within the Tester FPGA
    - Set the state of the RK05 bus RK11D mode signal (controls addressing mode in the drive)
    - Set the state of the RK05 bus Write Protect signal
    - Run a Tester DRAM test
    - Enter interface signal test mode (must be connected to an RK05 emulator to use this)
- Drive parameters are initialized to PDP-8 RK8-E values at power-up. Parameters for other drives can be selected by reading a disk image file from the microSD card. Disk image files contain a header that specifies all drive parameters.

Summary of Data Transfer Commands:

<table>
<tr><td rowspan="2">microSD</td><td></td><td>DRAM</td><td>← read sector
← read loop
← write loop verify →
write loop noverify →
← write loop zeroverify →
write loop zeronoverify →
write loop initialize →
← write loop random →</td><td rowspan="2">RK05
drive under test</td></tr>
<tr><td colspan="3">disk write [microSD-filename] →
← disk read [microSD-filename]</td></tr>
</table>

Drive Address Tests

- Enter drive address – **`addr [digit 0 to 7]`**
  Shortcut: **`a [digit 0 to 7]`**
  command "ADDR" or "A". Sets the drive address that the tester outputs on the RK05 bus. The RK11D/RK8E addressing mode is set by a different command.
- Address loop test – **`addr loop`**
  Output a sequence of device addresses on the RK05 bus. The sequence is BUS_SEL_DR_0_L through BUS_SEL_DR_3_L for RK05 mode and a 3-bit binary count sequence with binary encoded values (active low) on {BUS_SEL_DR_2_L, BUS_SEL_DR_1_L, BUS_SEL_DR_0_L}. The address selection sequence continues forever until a key is hit on the console.

Seek Tests

- Enter a cylinder seek list which can include "Restore" as a list item – **`makelist cyl`**
  Following the command, the tester prompts for a list of cylinders each separated by CR/LF. The list is used by the "**`Seek Loop`**" and "**`Seek Loopn`**" commands. The end of list is indicated by parameter value "x" or "X". The list can contain up to 50 values. Cylinder address values can range from 0 to 255. Note that 203 to 255 are invalid for the RK05 but these can be entered to test the invalid address detection capability in the drive. A cylinder address "r" or "R" can be entered which causes a Restore command to be sent at that point in the cylinder list.
- Display cylinder seek list – **`display cyl`**
  Displays the cylinder seek list that was entered using the "makelist cyl" command. The list can be edited by entering a new list.
- Loop on seek list – **`seek loop`**
  Send a sequence of seek commands on the RK05 bus. (Seek commands are bus strobe with a cylinder address or bus strobe with the Restore signal active.) Before seeking to the next cylinder, the tester will wait for the BUS_RWS_RDY_L signal to go active which indicates that the previous seek operation has completed. After seeking to the last cylinder in the list, the tester will loop back to the first item in the seek list and will loop forever. Looping terminates if the drive fails to acknowledge completion via the BUS_RWS_RDY_L signal within about one second, or if a key is hit on the console.
- Loop on seek list no verify – **`seek loopn`**
  Same as the seek loop command except that the tester does not wait for the BUS_RWS_RDY_L signal to go active before advancing to the next cylinder address in the list. The tester will not terminate the loop based on a failure to respond to the seek operation via the BUS_RWS_RDY_L signal. Looping stops when a key is hit on the console.
- Stepping seek loop test – **`seek step`**
  Begin seeking from cylinder 0 to the last cylinder and back to cylinder 0, incrementing or decrementing by one cylinder each time. Before seeking to the next cylinder, the tester will wait for the BUS_RWS_RDY_L signal to go active which indicates that the previous seek operation has completed. The cylinder address sequence for an RK05 is:
  0, 1, 2, 3….199, 200, 201, 202, 201, 200, 199…2, 1, 0, 1, 2, 3… etc.
  If the drive parameters indicate that the drive supports a different number of cylinders then the sequence will increment to the maximum cylinder address and back to 0. This sequence will

continue forever. Looping terminates if the drive fails to acknowledge completion via the BUS_RWS_RDY_L signal within about one second, or if a key is hit on the console.

Read & Write Tests

- Enter cylinder/head/sector list – **`makelist chs`**
  Creates a CHS list. Following the command, a list of cylinder/head/sector values is entered, with each value separated by a space with CR/LF after each list item. Each list item consists of three decimal numbers separated by spaces. Return or Enter following the sector parameter instantiates the CHS trio of parameters as a single list item. Multiple CHS trios can be entered, each trio on a separate line. List entry is terminated by entering the parameter value "x" or "X" as the cylinder with no entry for head or sector. A cylinder address "r" or "R" can be entered which causes a Restore command to be sent at that point in the cylinder list. The CHS list is used by the "**`Read Loop`**" and "**`Write Loop`**" commands. The CHS list can contain up to 50 values. Cylinder address values can range from 0 to 255. Note that 203 to 255 are invalid for the RK05 but these invalid values can be entered to test the invalid address detection capability in the drive.
- Display cylinder/head/sector list – **`display chs`**
  Displays the cylinder head sector list that was entered using the "makelist chs" command. The list can be edited by entering a new list.
- Read sectors in the cylinder/head/sector read list – **`read sector`**
  For all sectors in the CHS list, issue read commands to read those sectors from the drive under test and display the sector data in hex on the console. No data read verification is performed. This is primarily used for debugging the drive read functions.
- Loop on cylinder/head/sector read list, no verify – **`read loop`**
  For all sectors in the CHS list, issue read commands to read those sectors from the drive under test. No data read verification is performed. This is primarily used for debugging the drive read functions.
- Loop on cylinder/head/sector list, read/write test & verify, pseudorandom data – **`write loop verify`**
  For all sectors in the CHS list, initialize tester DRAM sectors with pseudorandom data. The data in each sector is unique. Write these sectors to the drive under test. Loop forever performing the following: sequentially read back CHS list sectors from the drive under test and compare the data with the previously initialized data in the tester DRAM. Write new pseudorandom data to the sector that was just read and advance to the next sector in the CHS list. Hit any key to stop the test.
- Loop on cylinder/head/sector write list, pseudorandom data, no verify – **`write loop noverify`**
  Similar to the "**`write loop verify`**" test except that no verification is performed. For all sectors in the CHS list: write pseudorandom data to these sectors in the drive under test. Loop forever writing new pseudorandom data to each sector in the CHS list. Hit any key to stop the test.
- Loop on cylinder/head/sector list, read/write test & verify, all-zero data with clock – **`write loop zeroverify`**
  Similar to the "**`write loop verify`**" test except that all-zeroes data is written instead of pseudorandom data. For all sectors in the CHS list, initialize tester DRAM sectors with all-zero

data. Write these sectors to the drive under test. Loop forever performing the following: sequentially read back CHS list sectors from the drive under test and confirm that the data read from the drive is all zeroes. Write new all-zero data to the sector that was just read and advance to the next sector in the CHS list.

- Loop on cylinder/head/sector list, read/write test & no verify, all-zero data with clock – **`write loop zeronoverify`**
  Similar to the "**`write loop zeroverify`**" test except that no verification is performed. For all sectors in the CHS list: write all-zero data to these sectors in the drive under test. Loop forever writing new all-zero data to each sector in the CHS list. Hit any key to stop the test.
- Initialize all sectors of the drive under test with zero data and a valid header word – **`write loop initialize`**
  For all sectors, initialize tester DRAM sectors with pseudorandom data. All-zeroes data is written to the data and CRC words of every sector. The Header word of each sector is initialized to the cylinder address shifted left by 5 bits, which is compatible with the DEC RK8-E and RK11-D/E controllers. Sector interleaving is used during this drive data initialization process to speed up the process of writing to the entire disk. The initialization process begins at cylinder 0. The operation ends after the sectors in cylinder 202 have been initialized. Write these sectors to the drive under test as the DRAM is being initialized.
- Pseudorandom cylinder/head/sector, read/write test & verify, pseudorandom data – **`write loop random`**
  For all sectors, initialize tester DRAM sectors with pseudorandom data. The data in each sector is unique. Write these sectors to the drive under test. Sector interleaving is used during this drive data initialization process to speed up the process of writing to the entire disk. Loop forever performing the following: choose a cylinder/head/sector using a pseudorandom selection algorithm. Read that sector from the drive under test and compare the data with the previously initialized data in the tester DRAM. Write new pseudorandom data to the sector that was just read and repeat by choosing a new cylinder/head/sector using the pseudorandom selection algorithm. Hit any key to stop the test.

Read & Write Disk Images

- Transfer a microSD file to a connected drive – **`disk write [microSD-filename]`**
  Write every sector in the drive under test by:
  1. Copy the disk image from the microSD card to the Tester DRAM.
  2. Copy all sectors from the Tester DRAM to the connected drive.
- Transfer from a connected drive to a microSD file – **`disk read [microSD-filename]`**
  Read every sector in the drive under test by:
  1. Copy the disk image from the connected drive to the Tester DRAM.
  2. Copy all sectors from the Tester DRAM to the microSD card.

Miscellaneous Commands

- Read back drive status – **`display status`**
  Displays a human-readable form of the important drive status signals and the global drive parameter values.
- FPGA Register direct read/write – **`reg <register address> <register data>`**
  Peek or poke tester SPI registers. Used for debugging unexpected problems.
- Set RK11D mode bus signal – **`mode rk11d <on/off>`**
  Controls the state of the RK05 bus signal: BUS_RK11D_L. When enabled (on) then binary drive select encoding is used. When disabled (off) then one-hot drive select encoding is used. The RK8E controller in the PDP-8 operates with rk11d mode off. The RK11-D controller in the PDP-11 operates with rk11d mode on.
- Set Write Protect mode bus signal – **`mode wtprot <on/off>`**
  Controls the state of the RK05 bus signal: BUS_WT_PROTECT_L. When wtprot is set to "on" then BUS_WT_PROTECT_L is active (low). When wtprot is set to "off" then BUS_WT_PROTECT_L is inactive (high).
- Interface Test Mode commands – **`mode itest <test name>`**
  Interface Test Mode is a special mode of the tester where the RK05 bus output signals can be controlled and monitored by the CPU by writing or reading SPI registers in the FPGA. Interface Test Mode enables an automated interface signal fault detection for finding interface signal path faults in testers and emulators. This is primarily used for testing boards after SMT assembly.
  - **`SCANOUTPUTS, SCANO, O`** – tester drives walking zeroes pattern on the RK05 bus signals. Emulator verifies the pattern is good. Verifies receivers and input path to FPGA.
  - **`SCANINPUTS, SCANI, I`** – emulator drives walking zeroes pattern on the RK05 bus signals. Tester verifies the pattern is good. Verifies drivers and output path from FPGA.
  - **`ADDRESS, ADDR, A`** – loops reading the address dipswitches and printing the value.
  - **`ROCKER, ROCK, R`** – loops reading the rocker switches and printing the value.
  - **`LEDTEST, LED, L`** – loops driving a walking pattern on the front panel LEDs.
- Emulator Board Version – **`boardversion <emulator board version>`**
  Enter the PCB revision of the emulator board connected to the tester. This is necessary only to properly run the scani or scano commands to achieve full test coverage of the I/O signals. Valid emulator board versions are: 0, 1, or 2. The default value is version 2, which is the most common of all emulator boards.
- Tester DRAM test – **`ramtest <hex start address> < hex number of bytes>`**
  Memory test of the tester DRAM. The address is a byte address. The physical word size is 16 bits.

The byte address range of DRAM is 0 to 0x1FFFFFF. Uses a pseudorandom pattern that is uncorrelated with the memory address. This toggles every memory bit and verifies the value. **`ramtest 0 2000000`** tests every memory location.

Note: after any interface Test Mode command ("`mode itest <command>`") is executed, the RK05 tester will operate only in Interface Test Mode. To function as an RK05 tester the unit must undergo a power-on reset. This is because the input/output signal configuration inside the FPGA has changed to operate in this special test mode and there is no easy way to restore the state without performing a power-on reset.

It is anticipated that the Interface Test Mode is only useful as a production-build verification test of the RK05 emulator, so it is reasonable to not recover gracefully from this mode.

Summary of commands with shortcuts

Field 1	Field 2	Field 3
ADDR, A	<digit 0 to 7>	
	LOOP, L	
MAKELIST, M	CYL	
	CHS	
DISPLAY, DISP	CYL	
	CHS	
	STATUS, S	
SEEK, S	LOOP, L	
	LOOPN, LN	
	STEP, S	
READ, R	LOOP, L	
	SECTOR, S	
WRITE, W	LOOP, L	VERIFY, V
		NOVERIFY, N
		ZEROVERIFY, ZV
		ZERONOVERIFY, ZN
		RANDOM, R
	ONCE, O	INITIALIZE, INIT
		INITADDR, IA
		SECTOR, S
DISK, DSK	READ, R	<filename>
	WRITE, W	<filename>
DIRECTORY, DIR, D		
REGISTER, REG	<register address>	
		<register data>
MODE, MD	RK11D	OFF, 0
		ON, 1
	WTPROT, WP	OFF, 0
		ON, 1
	CONTROLLER, CONT	RK8E
		RK11D
		RK11E
		ALTO
		<filename> *
	ITEST	SCANINPUTS, SCANI, I
		SCANOUTPUTS, SCANO, O
		ADDRESS, ADDR, A
		ROCKER, ROCK, R
		LEDTEST, LED, L
BOARDVERSION, BV	<emulator board version>	
RAMTEST, MEMTEST	<hex starting address>	<hex number of bytes>

* Not yet implemented as of Feb 2, 2025.

Notes:

Pseudorandom cylinder (0 to 202), head (0 to 1), sector (0 to 31) (CHS). Actual ranges are:

- Cylinder – 0 to numberOfCylinders – 1 (tester hardware limit is 8 bits)
- Head – 0 to numberOfHeads – 1 (tester hardware limit is 1 bit)
- Sector – 0 to numberOfSectorsPerTrack – 1 (tester hardware limit is 5 bits)

prbs is 0 to 65534 (use inverted feedback so all 1's is the lockout state)

Cylinder/Head/Sector values within the range are equally weighted: (prbs * max_value) / 65535

The Write Loop Random test functions as follows:

- Fill all tester DRAM sectors with pseudorandom data. Data in each sector is unique.
- Write all disk sectors sequentially (all sectors of tester DRAM to all sectors on disk). Sector interleaving is used to speed up writing the entire disk. (The sector interleaving algorithm works only for an even number of sectors per track.)
- Loop the following steps forever:
    - Use a pseudorandom generator to choose a cylinder/head/sector.
    - Save the tester DRAM data for the selected cylinder/head/sector in a CPU array.
    - Read that cylinder/head/sector from disk (disk to tester DRAM).
    - Compare the freshly read sector data with the data previously saved in the CPU array.
    - Compute new pseudorandom data for the sector location just read and write it to tester DRAM.
    - Write the new sector data to disk (tester DRAM to disk).

itest scan commands, Tester inputs, Emulator outputs, Hardware v0:

Tester input (hex)	Signal	Tester input pin	Emulator output pin
fffe	BUS_SEC_CNTR_L[0]	AH1	BL1
fffd	BUS_SEC_CNTR_L[1]	BH1	AP2
fffb	BUS_SEC_CNTR_L[2]	BM2	BK2
fff7	BUS_SEC_CNTR_L[3]	BR2	BJ1
ffef	BUS_SEC_PLS_L	AF2	BN2
ffdf	BUS_INDX_PLS_L	BL2	BM1
ffbf	BUS_DC_LO_L	AM1	BF2
ff7f	BUS_RK05_HIGH_DENSITY_L	BR1	BP2
feff	BUS_RWS_RDY_L	AM2	AH2
fdff	BUS_ADDRESS_ACCEPTED_L	AK1	AR2
fbff	BUS_ADDRESS_INVALID_L	AD1	AT2
f7ff	BUS_SEEK_INCOMPLETE_L	AL1	AS2
efff	BUS_WT_PROT_STATUS_L	AC1	BP1
dfff	BUS_WT_CHK_L	AF1	BK1
bfff	BUS_RD_DATA_L	AJ1	BS2
7fff	BUS_RD_CLK_L	AE1	BS1
ffff	BUS_FILE_READY_L	AL2	BN1

itest scan commands, Tester inputs, Emulator outputs, Hardware v1:

Tester input (hex)	Signal	Tester input pin	Emulator output pin
fffe	BUS_SEC_CNTR_L[0]	AH1	BL1
fffd	BUS_SEC_CNTR_L[1]	BH1	AP2
fffb	BUS_SEC_CNTR_L[2]	BM2	BK2
fff7	BUS_SEC_CNTR_L[3]	BR2	BJ1
ffef	BUS_SEC_PLS_L	AF2	BN2
ffdf	BUS_INDX_PLS_L	BL2	BM1
ffbf	BUS_DC_LO_L	AM1	BF2
ff7f	BUS_RK05_HIGH_DENSITY_L	BR1	BP2
feff	BUS_RWS_RDY_L	AM2	AH2
fdff	BUS_ADDRESS_ACCEPTED_L	AK1	AR2
fbff	BUS_ADDRESS_INVALID_L	AD1	AT2
f7ff	BUS_SEEK_INCOMPLETE_L	AL1	AS2
efff	BUS_WT_PROT_STATUS_L	AC1	BP1
dfff	BUS_WT_CHK_L	AF1	BK1
bfff	BUS_RD_DATA_L	AJ1	BS2
7fff	BUS_RD_CLK_L	AE1	BS1
ffff	TESTER_OUTPUT_1		BH2
dfff	TESTER_OUTPUT_2		BT2
bfff	TESTER_OUTPUT_3	AV1	BU2
7fff	BUS_FILE_READY_L	AL2	BN1
fffff	BUS_SEC_CNTR_L[4]		AB1

itest scan commands, Emulator inputs, Tester outputs, Hardware v0:

Emulator input (hex)	Signal	Emulator input pin	Tester output pin
fffe	BUS_CYL_ADD_L[0]	AK1	AR2
fffd	BUS_CYL_ADD_L[1]	AD1	AT2
fffb	BUS_CYL_ADD_L[2]	AL1	AS2
fff7	BUS_CYL_ADD_L[3]	AC1	BP1
ffef	BUS_CYL_ADD_L[4]	AF1	BK1
ffdf	BUS_CYL_ADD_L[5]	AJ1	BS2
ffbf	BUS_CYL_ADD_L[6]	AE1	BS1
ff7f	BUS_CYL_ADD_L[7]	AH1	BL1
feff	BUS_RK11D_L	AU1	BN1
fdff	BUS_STROBE_L	BH1	AP2
fbff	BUS_HEAD_SELECT_L	BM2	BK2
f7ff	BUS_WT_PROTECT_L	BR2	BJ1
efff	BUS_WT_DATA_CLK_L	AF2	BN2
dfff	BUS_WT_GATE_L	BL2	BM1
bfff	BUS_RESTORE_L	AM1	BF2
7fff	BUS_RD_GATE_L	BR1	BP2
ffff	BUS_SEL_DR[0]_L	AJ2	AH2

itest scan commands, Emulator inputs, Tester outputs, Hardware v1:

Emulator input (hex)	Signal	Emulator input pin	Tester output pin
fffe	BUS_CYL_ADD_L[0]	AK1	AR2
fffd	BUS_CYL_ADD_L[1]	AD1	AT2
fffb	BUS_CYL_ADD_L[2]	AL1	AS2
fff7	BUS_CYL_ADD_L[3]	AC1	BP1
ffef	BUS_CYL_ADD_L[4]	AF1	BK1
ffdf	BUS_CYL_ADD_L[5]	AJ1	BS2
ffbf	BUS_CYL_ADD_L[6]	AE1	BS1
ff7f	BUS_CYL_ADD_L[7]	AH1	BL1
feff	BUS_RK11D_L	AU1	BN1
fdff	BUS_STROBE_L	BH1	AP2
fbff	BUS_HEAD_SELECT_L	BM2	BK2
f7ff	BUS_WT_PROTECT_L	BR2	BJ1
efff	BUS_WT_DATA_CLK_L	AF2	BN2
dfff	BUS_WT_GATE_L	BL2	BM1
bfff	BUS_RESTORE_L	AM1	BF2
7fff	BUS_RD_GATE_L	BR1	BP2
ffff	BUS_SEL_DR[0]_L	AJ2	AH2
dfff	BUS_SEL_DR[1]_L	AK2	BH2
bfff	BUS_SEL_DR[2]_L	AL2	BT2
7fff	BUS_SEL_DR[3]_L	AM2	BU2
fffff	TSTR_SECT[4]_L	AV1	AB1