Troubleshooting & Diagnostics

5

Contents

Troubleshooting 5.2

Introduction

The CMI system relies on a complex interaction of hardware and software for proper operation. It is often difficult to differentiate between hardware faults, software bugs and operator errors. Before attempting repair of the Mainframe, establish that the fault is definitely a hardware fault in that unit. If in doubt, try the same series of operations on another system of the same revision level to ensure it is not a software bug and check with the User's Manual to ensure it is not operator error.

Due to the complexity of the system, this manual does not attempt to present an exhaustive list of faults and repair procedures. Instead, the detailed descriptions of the various components of the system should provide service personnel with a thorough understanding of the hardware. This knowledge should allow the general approach to fault finding outlined below to be adapted to isolating particular problems down to the board level.

Extensive diagnostic software is provided to thoroughly test most of the hardware. Of course, in order to run the diagnostics the computer section must be running at least to the stage of loading and executing the software. Use of the diagnostic software is described in detail in section 6 (below).

From time to time, due to the imperfect nature of the world and particularly during the early stages of production of a new machine, design faults emerge which chose not to reveal themselves during the prototyping and pre-production phases. The solution to these problems and the description of the conditions under which they occur are disseminated to all distributors in the form of Field Change Notices (FCNs). In general, FCNs do not need to be implemented on a particular machine unless the fault it cures is in evidence. The exception to this is when a hardware mod or ROM change is required for a new software release or a new hardware option to work properly - this situation would always be indicated on the FCN. Service technicians should always be familiar with the FCNs already distributed so that known problems can be readily recognised and fixed without delay.

A guide to troubleshooting the major components of the Mainframe follows:

Power Supply

There are two separate power supplies supplying a wide variety of voltages to different parts of the system so power supply failure may lead to anything from total system failure down to improper operation of a single component. Refer to the Power Wiring Diagram (at the end of this section) while reading the following description.

The CPU (comprising all circuit boards in the Logic Rack including and to the right of the Q256 System RAM) requires only +5V, +12V and -12V to function. These supplies come from the Switch-Mode Power Supply Unit (SMPSU) via the Logic Motherboard (CMI-35) and three LEDs are provided on the front panel of the CMI to indicate the state of each. The LEDs will not light if the relevant supply drops below two-thirds of the nominal voltage, however 4.8V is the absolute

minimum supply for correct operation of the CPU, so the LEDs are only indicators of gross failure. The best place to measure the +5V supply to the actual boards in the rack is across the 47uF electrolytic capacitors on the Waveform RAM cards, accessible with the lid removed from the machine. Note that the LEDs pick up their supplies via the ribbon cable connected to the Q133 card, so it is possible that a fault in the Q133 could cause the LEDs to malfunction.

Any other form of power supply failure will result in malfunction of specific parts of the system rather than the total system, as follows:

The Floppy Disc drive requires +5V and +24V from the SMPSU. There is no LED indicator for +24V, so a meter should be used to check this supply at the drive.

The Hard Disc(s), Streaming Tape and their associated controller boards are all supplied with +5V and +12V from the SMPSU, with supply conditions indicated by the front panel LEDs. Note that the LEDs only indicate the presence of the supplies on the Logic Rack, so meter checking is necessary to eliminate supply wiring faults between the SMPSU and the mass storage peripherals.

The VDU will only operate if it is receiving its 16VAC supply signal from the Toroidal Transformer, which is the first stage in the Audio Power Supply Unit. The transformer and CMI-310 Audio Power Supply board are mounted in the bottom of the machine to the right of the card cages and SMPSU, and are accessed by removing the bottom panel. One secondary winding of the Transformer is dedicated to the VDU supply which is routed through a fuse on the CMI-310 board then to the CMI317 Peripheral Connector board where a LED on the rear panel indicates the condition of the supply. This 16VAC supply is floating with respect to the rest of the system.

All fans in the system are supplied with 12V from the SMPSU.

The Music Keyboard and Alpha-numeric keyboard receive +20V and -20V unregulated from the CMI-310 Audio Power Supply. These supplies are fuse protected on the CMI-310 and go to the Keyboard socket via the CMI-317 Peripheral Connector where a LED on the rear panel indicates supply conditions.

The Audio Rack receives all its supplies from the CMI-310 except for digital +5V which comes from the SMPSU. Rev 3 CMI-310 modules include fuses for all audio supplies. Rev 2 and below include fuses for all audio supplies except digital +5V. A cable from the CMI-310 carries all audio supplies to LED indicators on the rear panel, mounted on the CMI-317 board. The Audio Modules, General Interface Module, SMPTE Module, and Mixer Module(s) depend upon the regulated +15V and -15V supplies and digital +5V. **The Stereo Sampling Module** has its own independent floating supplies: +20V, -20V and +8V all unregulated, so it is quite possible for a supply problem to cause only sampling failure. "Sample Fault" is the most likely system error message in this case.

The Headphone Amplifier uses +20V and -20V unregulated but is driven by audio circuitry which depends on the +15V and -15V regulated audio supplies.

In the event of a power supply failure, be suspicious that the failure may have been caused by a malfunction elsewhere, or incorrect setting of the mains voltage selector (on the Mainframe rear panel).

CAUTION

High voltages exist within the SMPSU (up to 700V) so servicing these units can be very hazardous.
Fan cooling of the SMPSU is vital and becomes ineffective if the cover over the unit is removed. For safety also, do not operate the CMI with the SMPSU cover removed.

CAUTION

Applying 220V or 240V when the mains voltage select switches are set to 110V will usually destroy the SMPSU.

If power supply failure is traced to the Switched Mode Power Supply Unit, replace the complete unit and return to Fairlight Instruments.

Computer (CPU) Section

A failure in the computer section may result in permanent or intermittent malfunction of the system. Normally a Boot (floppy) Disc is used to start the system which transfers immediately to the Hard disk for loading of the rest of the system software. If this does not complete successfully, try first to boot from a Floppy-only CMI System Disc. This will eliminate the Hard Disc system (both the software on the Hard Disc and the actual hardware) as source of the problem. If you are used to the speed of Hard Disc operations, do not be alarmed by the slow operation of the Floppy-only System.

Failing this, try to load a Diagnostics system floppy disk. This uses the QDOS operating system (same as Series IIX) which is much simpler than the Series III OS-9 System and less demanding on the hardware. If this is successful, refer to section 5.7 which describes how to approach non-fatal digital faults.

Fatal Digital Faults

If the system will not run at all (will not load the diagnostic disk) start by checking the following:

1) ENABLE/SAFE switches on the front panel must be in the Down (RUN) position.
2) Mains voltage selector on the rear panel must be set correctly.
3) Power must be applied, as indicated by the three LEDs on the front panel, and ten LEDs on the rear panel.
4) The System Disk must be in good order (as indicated by testing in another system) and inserted correctly.
5) The floppy drive containing the System Disk should be configured as Drive 0, and the switch on the front of the QFC-9 card in the mainframe should be set in the raised position. (It is easy to accidentally flip the switch when removing the floppy disk ribbon cable.)

If these items are all in order, but the system disk will not load, confirm that the fault lies in the Mainframe by disconnecting everything except the AC supply from it and trying to load the disk again. If successful, find out which connected peripheral is interfering with system operation by trying one at a time. Otherwise, attempt to ascertain whether the CPU is running but unable to access the disk, or the CPU is failing to execute at all. The LOAD SYSTEM DISK IN DRIVE message on the VDU indicates immediately that the CPU does get as far as executing the startup software contained in the CPU Control Card ROMs, so the fault probably lies somewhere in the floppy disk sub-system (see Sec. 5.4). The absence of the message means either the graphics sub-system is not working or the CPU is dead. Even if neither the Floppy disc nor the VDU is operational, a clue to the operation of the CPU can be gained if the Parity Error LED on the front of the Q256 RAM turns on at power up then off again after about 2 seconds, since the clearing of the parity error register is under software control of the Q133 boot ROMs.

If the disk will still not load, the system should be "stripped down" until operation is restored. Remove cards in the following order, attempting to load the disk after each stage:

1) Remove all Waveform RAM cards (CMI-39), all channel cards (CMI-31), Channel Support card (CMI-32),.Waveform Processor (CMI-33), General Interface (CMI-28). With this configuration, the Hard Disk will not operate and SCSI diagnostics will fail, but it is still possible to boot from the floppy.
2) Remove the second System RAM card, if installed, from slot 20.
3) Remove the SCSI Controller (Q777) from Slot 27. Close the PCB link leading to edge connector pin 71 on the wiring side of the QFC-9 board (see "potential red herrings" below).
4) Remove the Graphics card (Q219) from slot 25.

5) Replace the Graphics card and remove the Floppy controller (QFC9, slot 26). Obviously it will not be possible to load the system disc but if the CPU and graphics sub-systems are functional, a "* READY *" message followed by the Processor 2 6809 monitor prompt ":" should appear.

If the fault still appears to be in the CPU itself, substitute the CPU cards with known good spares, one at a time until all have been replaced. If the fault persists, the problem must be in the power supply, floppy disk system, motherboard or cabling. Refer to Section 7 (Motherboard Signals) and Section 8 (External Connections) below, for details of motherboard and external signals.

CAUTION

Potential "red herrings" to fault finding by board elimination

Daisy chaining of DMA (Direct Memory Access) signals means that some boards will not work unless certain other boards are present in the system. Up to Rev. 3 of the CMI-35 motherboards, the order of dependency is:

0. SCSI Controller (Q777)
1. Floppy Controller (QFC9)
2. General Interface (CMI-28)
3. Waveform Processor (CMI-33)

Each item in the list must have all higher items present in the system in order to work. For example if the CMI-28 is absent, the CMI system will abort its startup, and diagnostics involving the Waveform Processor will output "WP not present or won't load" messages because the WP is unable to communicate with the CPU unless it gets its daisy chain signals from the CMI-28. The system can be run without the SCSI Controller by using the $\overline{\text{EDL}}$ output from the QFC9. On Rev 6A and below, this involves repairing the cut in the track going to pin 71A of the edge connector. On later revisions a link option connecting to the same place must be closed, but this link must be reopened or the track recut to install the Q777.

Provision has been made for future systems with SCSI-compatible floppy disk drive or no floppy at all. Linking pins 71A and 72A on the Q777 will result in a new order of dependency:

1. SCSI Controller (Q777)
2. General Interface (CMI-28)
3. Waveform Processor (CMI-33)

Refer to the Digital Motherboard Section 2.15 for further detail.

Floppy Disk System

The Floppy Disk system consists of three sections - the floppy-disk controller card QFC9, interconnecting cables, and the floppy-disk drive. More than one drive may be installed if required.

A fault in the controller card or cabling will generally result in hard disk errors or total failure to access disks. Soft or intermittent disk errors will usually be caused by a faulty or misaligned drive.

A toggle switch mounted on the front edge of the controller card reverses the drive select signals to the drives so that the logical drive numbers can be swapped for diagnostic purposes, although this is obviously useful only if more than one drive is installed. For example, if a disk error is reported during boot-load, the drives can be swapped (switch down) and the system disk inserted in the second drive. If it then loads sucessfully, the first drive is faulty. If the fault persists, the fault is in the controller, cabling or power supply, unless the system diskette itself is faulty.

Refer to the Floppy Disk maitenance section for full details of disk drive maintenance.

Hard Disk System

Refer to the Mass Storage Devices Section 5.

Non-Fatal Digital Faults

Non-fatal faults encompass those which do not prevent the CMI from loading and running at least the Diagnostic System disk. Such faults therefore include minor faults in the CPU section and nearly all faults in non-CPU cards except those which cause the CPU bus to be corrupted or by some other means crash the CPU. Once the Diagnostic System disk is able to be loaded, faults may be isolated by the intelligent use of diagnostic software, board swapping, and observation of the behaviour of the machine under the full CMI System software if it is possible to load that as well.

Troubleshooting using the CMI System Software

When the CPU is starting up the CMI System software it checks to ensure that all peripheral processors (General Interface processor, Waveform Processor, Channel processors) are working. If either the GIF or Waveform processors do not respond correctly the startup procedure is aborted with a message on the console, so these problems are easily detected. Both LEDs on the GIF and WP cards are on after power-up, turn off when the processors are started by the CPU, and one or both come on again if the System load aborts. LED functions are explained later.

When starting up Channel Processors, the CPU first does a quick test on each Channel's on-board memory to see if the Channel is installed (and working). It then loads Channel driver software into the memories of the channels present, releases the Channel Processors

from reset (RUN LED switches on), then checks for sensible communication with them. If a Channel Processor, whose memory test worked, does not respond, the startup procedure is aborted with a message. The usefulness of abort messages is a function of software release level and will become more informative in later releases. Messages mentioning "mptask" and "wptask" refer to the General Interface and Waveform processors respectively.

If the CMI System software loads successfully it can be further used to investigate digital faults. For example, if the sequencers play music successfully but notes played on the keyboard have no effect, the General Interface card must be 95% working since it has a major role in running sequences. The fault may lie somewhere in the path from the keyboard to the GIF card - the General Interface Support module in the Audio Rack, the I/O interface of the GIF card, interconnecting cables, or the keyboard itself (see MIDI Frame Display section below). Another example is the case of distorted or crackly waveforms - a problem which is common because there are so many possible causes in both the analog and digital sections of the machine. The following clues can be used to isolate the problem:

i) If the clicks or crackles always sound the same between successive key presses or between successive loads of the same sound then the problem may be in the sound sample data (several sounds in the first release of the Fairlight Library were prone to this problem).

ii) If it is observed that placing the sound in different places in Waveform Memory (by sampling or loading other sounds first) causes the problem in only one place in memory, then it is highly likely that the Waveform RAM card corresponding to that area is at fault. Do not jump to conclusions too soon though, as the Waveform Processor or a Channel card may be having problems addressing that area.

iii) If it is found that sampled sounds are OK but sounds from disk are bad, then the Waveform RAMs and Channels are innocent but suspect the Waveform Processor, the SCSI controller, the disk drive, and the voice data itself on disk.

iv) If a defective voice is overwritten with an internally generated sine wave which then plays without problem, then the SCSI controller, disk drive, or voice data are suspect for sounds from disk, or the Sampling Module is faulty in the case of sampled sounds. The Waveform Editor is very useful in examining whether waveform errors are in Waveform RAM or only occur when the sound is being played.

MIDI Frame Display

It is possible to generate a display of raw MIDI data received by the CMI. If, for example, sounds can be played from the sequencer but playing on the music keyboard has no effect, the display can verify whether any data is being received from the keyboard. The operation

of controls such as modulation wheels and switches can also be checked. Future software releases may have a built-in command which produces a MIDI frame display without the complicated procedure above. Refer to the user manual to find out if this command is available.

To obtain the display without a special command, boot the CMI system normally then follow the procedure below. System prompts and messages are in italics, you type the text in bold. Explanatory notes are on the right in normal print.

$	Call up an OS-9 Shell
Shell	
# com68 -r -i -m	Run program to talk directly with GIF processor
68k Communications Program v2.8 - type '?<cr>' for help	
<cntrl-N>	Interrupt GIF processor
.	
.	3 lines of 68000 internal register dump
.	
K: **80200**/*00* **ff<cr>**	Set software flag for screen output
K: **p**	Instruct GIF processor to proceed

From this point each incoming MIDI data frame relating to a key or control movement etc. will be printed on the screen and will look something like

pQ:01903c4f

The data above is what you get if you depress middle C with a moderate key velocity. If anything at all gets printed then the music keyboard and all the connections between it and the GIF card are obviously working so the actual data does not need careful attention. Just in case you are interested however, the "pQ:" means the data is in the "Input Play Queue", the following "01" means it is going to operate on instrument number 1, and the last six characters are the hexadecimal values of the 3-byte MIDI frame.

When you have finished looking at the MIDI frame display, return to normal operation as follows -

<cntrl-N>	Interrupt GIF processor again
.	
.	3 lines of 68000 internal register dump
.	
K: **80200**/*ff* **0<cr>**	Set flag back to normal
K: **p**	Proceed
<cntrl-C>	
command: **x<cr>**	Exit communications program
# <ESC>	Exit shell, return to CMI system.

Channel Allocation Scheme

In the process of testing using the System software, it is often desirable to play a certain channel or channels. This requires an understanding of the channel allocation scheme. When creating voices, the lowest unused channels are used. When playing a voice with Nphony greater than one, the longest unused channel is played (note that this is very different from the Series I/IIX scheme). Channels equally unused are cycled through numerically.

Thus to test all channels in succession, create a voice (the in-built sine voice will do for go/no-go tests) with an Nphony of 16. Playing a single note repetitively, careful to release each note before beginning the next, will then cycle through in numerical order. Playing a chord will permanently upset the order, although all channels will still be cycled.

To test a single channel continuously, create a voice with sufficient Nphony to occupy all lower channels, then create a voice with Nphony 1 for the test channel and select that voice to be played by the keyboard.

Avoid using sounds with velocity sensitive level controls for this kind of testing, or turn Keyvel control off, otherwise unconscious variations in key velocity will cause confusing differences between channel levels.

Using the CMI System software for troubleshooting is not as exhaustive as running diagnostic chain tests but is often much quicker, and the shrewd technician with a full understanding of the function of each component in the system can use it to narrow down the fault before turning to the specific diagnostics. LED indicators on the front of many boards have been included also for this purpose. The rest of this section briefly describes the kinds of digital faults each non-CPU card can produce and the function of the LED indicators.

General Interface Card - CMI System load abort, failure to play notes from the keyboard, failure to play sequencers, improper operation of SMPTE and metronome.

Note: if the 26-way MIDI/SMPTE ribbon cable which connects from the Audio Motherboard to the front of the General Interface card is unplugged or the CMI-332 MIDI module, excessive spurious interrupts to the GIF processor may cause the sequencer to slow down, operate intermittently, or stop completely. It is therefore advisable to always have the CMI-332 plugged in and the cable securely attached.

LEDs: Processor HALT and RESET. HALT is nearer the front. Processor is halted when HALT light is on; reset when both are on.

Waveform Processor - CMI System load abort, bad waveforms in memory, sampling problems.
LEDs: Processor HALT and RESET. HALT is top LED. Processor is halted when HALT light is on; reset when both are on.

Channel Support Card - Bad waveforms in memory, correct waveform in memory but bad waveform data reaching audio modules, failure of one or more Channel card to operate properly or at all, incorrect or absent control voltages from all Channel cards to Audio Modules.

If Channel Card operation faults remain the same when Channel cards are swapped around or replaced with known good ones, then the Channel Support is most likely to be at fault, although faulty edge connector or motherboard is possible.

Channel Cards - CMI System load abort, failure to play certain notes, bad waveforms from one channel, corrupted waveforms in memory.

LEDs: Four, from top, "A" side playing, "B" side playing, Channel processor RUN (ON = not reset), FIRQ (Fast Interrupt Request from Channel Support; ON = requests being serviced). The power-on state of the FIRQ LED is undefined but it should always be on when the RUN LED is on.

Note that a faulty channel may cause a fault which appears to be a particular Waveform RAM card unless different sounds are loaded to make the channel play in more than one 2 megabyte space.

A faulty Channel card may cause other channels, especially those to the left of it, to also appear faulty due to daisy chaining of Waveform Bus allocation signals from one channel to the next. Any channels to the left of a gap will be unable to generate sounds.

Failure of both "sides" of a channel card to play may be due to the Channel, the Channel Support card, or the GIF but failure of only one side is almost certainly the Channel card itself, an Audio module problem or a faulty cable between the two.

Waveform RAMs - Bad waveforms in memory.

If it is suspected that a particular RAM card is at fault, swap with a known good card or change the assignment of RAM card numbers using the on-card DIP switches. If the bad waveforms still occur on the same card but with different sounds and different addresses, the fault is in the RAM card itself or the motherboard slot. Move the RAM cards around in the last 7 slots of the rack to eliminate the latter. The physical position of Waveform RAM cards is unimportant to the system since space assignment is achieved with the 4-pole DIP switch. By convention, however, cards are placed consecutively, starting with Card 1 nearest the channel cards.

Trouble Shooting using the Diagnostic Software

The diagnostics and their use are fully covered in a separate section. It suffices to state here that the diagnostics can be used individually to find more information about a well-localised fault or chain tests can be run to test the whole system automatically. Chain tests are also available to facilitate continuous testing of specific parts of the system.

The Transition from Digital to Audio

If all diagnostics pass and the CMI System software appears to operate correctly yet sounds are not being played correctly it would seem reasonable to proceed to the Audio Rack to investigate the fault. But first it is necessary to ensure that all signals flowing from the Digital Rack are reaching the Audio Rack, since it is not possible for the CPU to internally test the interface components or the cables between the two racks.

Audio Module Inputs

First attempt to isolate the fault as a Channel card or Audio Module problem by systematic swapping, preferably with known good boards or with other boards in the system. If the Channel card is eliminated, the problem must be either in the Audio module, power supply, or the interconnections.

Each Audio Module receives digital waveform data from the Channel Support Card, and Data Clocks, Pitch Clocks, and Control Voltages from its corresponding Channel Card. These signals are most easily checked by placing the Audio Module on an extender card and using an oscilloscope. Refer to the Audio Module schematics for where to look for each signal. Waveform data is bussed along all modules so check it first if no modules work, by playing any note and examining the changing data on the outputs of the differential receivers. The clocks and control voltages appear only when the corresponding Channel Card is playing (make sure you are looking at the correct half of the Audio Module).

If any input to the Module is missing or incorrect, check back along the Audio Motherboard, ribbon cables, to Channel Card or Channel Support output buffers.

Mixer Controls

The digital control signals for the Mixers can be checked automatically by replacing the cable from the Q133 to the Audio Motherboard with a PIA shorting plug and running the PIA test in the DBTST diagnostic (see Diagnostics section). If this passes, the problem must be in the cable, the Audio Motherboard, or the Mixer itself. The latter can be eliminated by board swapping.

Stereo Sampling I/O

The Sampling Module receives control data and sends digital waveform data and clocks via an optically isolated serial link to the Waveform Processor. The "ADC timeout" message from the CMI System software indicates that a Start Conversion pulse was sent to the Sampler but no End-Of-Conversion pulse came back.

The opto-isolated link is a current loop system so it is difficult to see any signals on the cable. Use an oscilloscope to check output signals on the open collector gate inputs, and input signals on the opto-isolator outputs on both the CMI-337 and CMI-33. Refer to the schematics in both cases.

Waveform Diagnostic Card

Fairlight Instruments will make available to Service Centers a Waveform Diagnostic Card which will intercept all outputs from the Digital Rack to the Audio Rack and interface them back to the CPU. This will allow automatic testing of all the above signals and hence positive isolation of any fault between the two racks. Full documentation of the card plus diagnostic software will be available when deliveries begin.

Audio Faults

In general, if a fault occurs in the Audio rack, it is easy to isolate which module is responsible. This section briefly describes some of the more common audio problems which can be fixed without resort to circuit board exchange.

CMI-331 Audio Module

Distorted or "crackly " waveform from DAC output - may be caused by damaged DAC (MP7616 DACs are severely static sensitive) or faulty logic circuitry preceding the DAC. (The DAC output is current mode, so its output must be observed at the output of the current-to-voltage op-amp circuitry). Errors in the most significant bits of the DAC or its driving logic can easily be seen on an oscilloscope, while errors in the least significant bits appear as distortion components on the output of a distortion meter.

If the DAC output is correct but the VCF produces nothing but a buzz, recalibration of the VCF is required. Too high a Q setting can result in latch-up of the VCF with occasional negative spikes. Incorrect frequency adustment of the VCF will cause sounds to be duller than they should be, or allow digital "grit" to get through to the output. Refer to the diagnostics description for the calibration procedure.

VCA miscalibration can result in incorrect output level and clipping. Serious miscalibration can even cause an apparently square-wave output from a sine waveform. Again, refer to the diagnostics description for the calibration procedure.

Cross-talk between the two channels on each audio module or strange output levels can be caused by incorrect connections to the audio outputs of the machine. All audio channel and mixed outputs are balanced, with a low output impedance (typically 33 ohms). If driving single-sided lines, leave the unused outputs open.

CMI-334 Mixer Module

Clipping of the mixer output can occur if multiple channels are played with highly coherent waveforms. The gain of the mixer is set so that a reasonable level is produced when only one channel is playing. However this means that clipping will occur if all 16 channels play, for example, in-phase sine waves. A -20dB cut switch is mounted on the mixer module, accessible from the rear panel, to avoid this problem. The switch has no effect on any other outputs.

The mixer output may appear to be rather noisy when referenced to one channel driving it - a typical SNR in this situation is -67dB. However with reference to the mixer's full output of +4dBm, achieved by having several channels driving it in phase, just below the onset of

clipping, the SNR should be more like -90dB which will be the sum of all the individual audio module noise levels. The -20dB cut switch will also reduce the absolute noise level by 20dB so that the SNR will remain the same.

CMI-337 Stereo Samping Module

Poor performance of the ADC at high sampling frequencies (above 50kHz) can result if the 20kHz brickwall filter is not switched out. Software versions 2.03 and beyond take care of this automatically by preventing too high a sample rate.

Revisions 1 and 2 of the CMI-337 required trimming of the DC offset to the ADC input. Refer to the detailed information on the CMI-337 for the calibration procedure.

Glitches around the zero crossings of very low level signals, and partial or complete failure of the digital timing circuitry can also result from miscalibration of the module.

Since the Sampling Module has its own independent (and floating) power supplies, complete failure may also be the result of a power supply problem. Even if the LED indicators on the rear panel are lit, check that the supplies are actually getting to the module.

Several known problems which occured on early revisions of Audio Rack modules have been the subject of Field Change Notices. To avoid re-inventing the wheel at best, and at worst, much frustration and customer grief, always check the FCNs as soon as the basic symptoms of a fault have been established.

SCSI Interface

All mass storage (excluding the floppy system) is accessed through the Q777 SCSI adapter which connects to various controllers via a 50 way ribbon cable. A failure in this card will render all SCSI devices inoperative. Normally a system will have a hard disc and a tape unit attached to the SCSI buss, so after booting from a floppy disc try to access both these devices. TP DIR with a good tape will soon show whether the tape system is working. If no SCSI device appears to work then check the following.

1. Thoroughly check SCSI cable.
2. Check all attached controllers have power.
3. Check external SCSI conection.
4. Replace or run diagnostics on Q777.

Hard Disc System

If the Hard Disc system is faulty, it may not be possible to access any files through OS9, or there may be a fatal error during operation. The problem could be a controller fault, drive fault or simply a cable problem.

First try to eliminate simple faults by checking all cables associated with the disc unit :-

* power to drive
* power to Adaptec
* SCSI to Adaptec
* both cables from adaptec to drive

Next, replace the Adaptec controller (if possible) and retry.If the fault persists then the Hard disk unit is faulty and must be replaced.

Streaming Tape

As with the Hard disk check all cables and power supplies. The tape controller card has a green LED visable from the rear and if the controller is functioning this LED will flash continuously. The tape transport operation may be tested by inserting a tape into the drive and closing the door. The tape should reposition, the heads should move and finally the red LED at the front of the tape unit should come on. Any radical departure from this behaviour suggests a faulty tape unit.

If the Tape system operates but returns "TAPE ERROR 3", then check the following :-

* faulty tapes
* dirty heads

If "TAPE ERROR 3" persists then the tape drive requires replacement.

Diagnostics 5.3

Introduction

This section describes the set of diagnostic test programs available on the Diagnostics Disk for testing and debugging all the plug-in circuit modules of the Fairlight CMI (Series III).
Conventions used in this documentation:
$nnnn = hexadecimal notation (where n = 0 to F)
[ret] = return key, used in terminating most commands.

Running the Diagnostic Disk

The diagnostics run under the QDOS operating system. They are therefore supplied on a QDOS system disk which should be loaded into the CMI instead of the usual CMI system disk immediately after power-up or RESTART.

As soon as the disk drive door is closed, the CMI will load QDOS automatically and display a sign-on message giving version and revision numbers.

The QDOS prompt will then be displayed. This is just an equals sign "=" on a line of its own. This indicates that the computer is ready to accept a command.

For further details of the QDOS operating system features, refer to the QDOS User's Guide.

Chain File Diagnostics

Complete diagnostic tests may be run on basic CMI hardware by using chain files. Chain files effectively combine all the individual test programs relevant to a major test. This eliminates typing in each individual test.

Chain files are invoked by typing CHAIN followed by the name of the chain. For example, to test the whole system, type CHAIN TEST [ret].

The following is a list of the chain files currently in use.

CHAIN	FUNCTION	NOTES
TEST	checks most of the system firstly with overall interrupt tests then tests the Q133 debug card, the Q219 graphics card, the Q256 RAM cards, the CMI33 waveform processor, the seven CMI39 waveform RAM cards, and the CMI28 general interface card. Some general CMI31 channel card tests and finally some disc drive tests.	
WAVERAM	checks the waveform RAM cards for memory errors with channels looping	

See Chain Tests for more details about theses chain files.

Running the individual Test Programs

Most of the diagnostic programs make use of a common command interpreter for operator control, so there is a uniform command syntax employed throughout, and several test options available as standard. Tests may be run by typing

<test name>[,<option1>,<option2>...,<optionN>] [ret]

where the <test name> is as described in each section. Options are of the form

<O>=n where <O> is a single character and n is an integer.

Standard options are;

P=n Repeat test command n times. Using "C" instead of an integer initiates continuous testing.

N=n Select test number n. There are usually several test commands with the same name. By default, all tests are executed sequentially but single tests or subsets of the available tests can be specified.

For example	N=1	Test no. 1 only
	N=1,3,5	Tests 1, 3 and 5
	N=4-7	Tests 4 to 7 inclusive

Some tests, which require a waveform to be observed, wait for the spacebar to be pressed before terminating or proceeding to the next test.

To obtain a reminder of what tests are available from the current test program being run, type

LIST [ret]

To repeat the last test, just type

R [ret]

If an error condition occurs, a moderately helpful message is printed on the console and the program returns to the command interpreter or continues testing depending on the setting of the errors option.

Some tests have error handling options which allow testing to continue with or without error messages being displayed.

Successful tests terminate without comment and return to the interpreter or proceed to the next test as soon as completed. Certain tests require the user to check waveforms with an oscilloscope and will not terminate or proceed to the next test until the spacebar is pressed.

Few diagnostics are available for testing this card (Q-133) since it is not possible to load the DOS and run a test without most of the card functioning correctly, some peripheral functions are tested by the program DBTST.CM, which may be run by typing

DBTST [ret]

with the CMI diagnostics disk in drive 0.

The standard command interpreter is used so the LIST and R commands, and P and N options are available as described in the general introduction. Tests are run by typing

<test name>[,<option1>,<option2>...,<optionN>] [ret]

User Peripheral Interface Adapter

The PIA tests require a special test plug to be inserted in the user PIA socket (the one nearest the top of the board) which has the effect of connecting the A side of the PIA to the B side. Connections are as follows:

1 - 24, 2 - 25, 3 - 26,5 - 22, 6 - 21, 7 - 20,
8 - 19, 9 - 18, 10 - 17,11 - 16, 12 - 15.

Test Name: PIA
No. tests: 6

Test No.1 *PIA B/Send A/Receive*
Test No.2 *PIA A/Send B/Receive*
Purpose: With the test plug connecting the two sides of the PIA, A should be able to write to B and vice-versa.

Both tests check each side of the PIA individually first by defining the side as all bits inputs, changing them to outputs, then writing 0, $FF, $FE etc. down to 0 again to the data register and reading back to verify each write.

Test no. 1 then sets side A as all inputs and side B as all outputs and writes all values from 0 decrementing back to 0 to side B, AND reading from side A. Test no.2 does the same in the opposite direction.

Test No.3 *CB2/Send CA2/Receive -ve*
Test No.4 *CB2/Send CA2/Receive +ve*
Test No.5 *CA2/Send CB2/Receive -ve*
Test No.6 *CA2/Send CB2/Receive +ve*
Purpose: Interrupt inputs/control outputs check.

The signal specified by "Send CX2" is configured as a control output whose state is determined by CRB-3 in the PIA control register. The other signal is configured as an interrupt input which will set the interrupt flag in the control register on an edge whose direction is indicated by the "Receive +ve or -ve".

The transmit end is first set to the state which will allow the wrong transition to cause an interrupt, (i.e. if the interrupt receiver is +ve edge triggered, the transmit end is set high) and the flag is checked as clear. Then the transmit state is toggled and checked again as still clear. A second toggle should trigger the interrupt flag. The actual IRQ output is disabled during the test.

System Timer

Test Name: TIM
No. tests: 2

Test No.1 *System Timer Read/Write Latch*
Purpose: Check ability to communicate with timer.

The 3 timers in the 6840 timer are put into the preset state and all numbers from zero to $FFFF are written to the timer 1 latch Each write is followed by a timer read for verification. Timers 2 and 3 are tested in the same way. Each write/read is a double byte transfer through the 8-bit bus.

Test No.2 *System Timer Internal Clock Timeout*
Purpose: Check timeout operation under internal clock.

The timers are clocked by the internal clock. All three timers are initialised to $FFFF then started. A software timing loop is used as reference, and the timer status is continually polled for premature timeout. Timeout must occur within a certain tolerance before or after reference timeout. Clock outputs are enabled during the tests.

Interrupts

Test Name: INT
No. tests: 1

Test No.1 *Interactive Interrupt Test*
Purpose: Test interrupts.

Enable interrupts, then display any which occur. Exit when <ESC> typed. Interrupts may be induced by user poking interrupt lines with an earthed probe.

Asynchronous Communications Interface Adapter

Test Name: ACIA
No. tests: 1

Test No.1 *Comms ACIA Send/Receive 19200 Baud*
Purpose: Tests asynchronous transmit/receive.

Checks that DCD and DSR status bits follow RTS, character is sent and received, interrupt flag set after character is sent, parity error and framing error. This test requires a 10-way loop back plug to be inserted.

Test Name: LATCH
No. tests: 1

Test No.1 *Comms Latchup Protect*
Purpose: Test if ACIA latched up (usually on power up).

Transmits a character and goes into a software timeout loop. Error message displayed if character not received.

Lightpen/Graphics Card Diagnostics
Use Light Pen/Graphics test, LGTST.CM by typing

LGTST [ret]

with the CMI diagnostics disk in drive 0.

The standard command interpreter is used so the LIST and R commands, and P and N options are available as described in the general introduction. Tests are run by typing

<test name>[,<option1>,<option2>...,<optionN>] [ret]

Light Pen Timers

Note: Although the Series III does not use a lightpen as a graphics input device, the circuitry is still present and facilities such as the 6840 timer may still be used for general software purposes. Therefore the following diagnostics should be used. Either processor can run the TIM tests, as selected by the SEL command (see below).

Test name: TIM
No. tests: 4

Test No.1 *Light Pen Timer Read/Write Latches*
Purpose: 6840 timer preset latches can be written to and the counters read.

Timers are put into preset state in which the counters always reflect the contents of the preset latches. Then each timer is write/read tested with all numbers from 0 to $FFFF. Each write/read is a 16-bit transfer through the 8-bit bus.

Test No.2 *Light Pen Timer Internal Clock Timeout*
Purpose: Correct timeout from timers under internal clock.

The internal clock is provided by the BCA signal. Timer outputs are enabled and latches preset to $FFFF. Then all three counters are released and their timeouts compared to a software status-polling (to sense timeout) timing reference loop.

Test No.3 *Light Pen Timer 2 External Clock*
Test No.4 *Light Pen Timer 3 External Clock*
Purpose: Timers under external clock

Timer 2 counts frames, thus gets a 20mS clock cycle. The test is not synchronised to the frame pulses so a +/-10mS jitter is permissible. The timer is preset to count 200 clocks (4 secs), then released and compared to the software reference with the required tolerance.

Timer 3 is clocked by processor 2 phase 2 (1MHz). It is preset to $FFFF then released and its timeout compared to the software reference.

Both timers run in single shot mode with outputs enabled.

Light Pen PIA

Test name: PIA
No. tests: 1

Test No.1 *PIA Test*
Purpose: The PIA is used to control the scroll register as well as the light pen circuitry.

Each side of the PIA is configured as all outputs then all numbers from zero backwards down to zero are written to the data latches and verified.

Processor Access Selection

Test name: SEL
Purpose: Allows user to specify which processor runs the TIM tests.
Options: C=n CPU selection
Range 1-2, default 2

In addition to selecting which processor runs the TIM tests, the SEL command adjusts the timing parameters used to compare the hardware timer to software loops. This is necessary because refresh occurs on P1 cycles, resulting in P1 loops running slightly slower.

Video RAM Testing

Test name: VRAM
No. tests: 8
The VRAM tests do not use the SEL command to select processors. Which processor runs these tests is determined by the default graphics processor select byte contained in the Q133 boot ROM (normally P2 for the CMI).

Test No.1 *VRAM Processor*
Purpose: This determines which processor has access to the video RAM. To actually change processors, use the SEL command.

Test No.2 *Random VRAM Test*
Purpose: Random numbers are written to video RAM, then compared with original. Error if not the same.

Test No.3 *AA VRAM Test*
Purpose: Video RAM filled with hex number AA.

Test No.4 *55 VRAM Test*
Purpose: Video RAM filled with hex number 55.

Test No.5 *XY Byte/Inc*
Purpose: Test XY vector drawing hardware by writing random numbers and auto-incrementing. Numbers are then compared with actual video RAM contents.

Test No.6 *XY Byte/Dec*
Purpose: Same as XY/Inc except vector drawing hardware auto-decrements.

Test No.7 *XY Draw*
Purpose: Test the XY vector hardware.

This test draws a 2 byte wide by 16 byte high pattern into video RAM using the XY vector hardware, then checks that it was correctly drawn in the correct absolute locations in memory.

Test No.8 *Random XY. Byte Read*
Purpose: Video RAM is directly filled with random numbers. Random numbers are then regenerated and compared with video RAM via XY hardware.

Q256 memory card memory tests

When testing system memory it is desirable to have as little memory space as possible taken up by the operating system and the memory test itself. In order to restrict the memory diagnostics within a 16K block, they are split into three separate programs: **MEM256**, **MAP256** and **DMA256**. The first performs bulk memory testing in 16K chunks and exercises the parity generation/checking system. MAP256 tests the full capability of the memory management system, except for the automatic DMA map selection, which is tested in DMA256.

A fourth program, MEMDBG is a small and simple program which exercises specific sections of the Q256 circuitry in very tight loops and thus generates the stable CRO traces needed to find circuit malfunctions. It is only appropriate to use MEMDBG once MEM256 or the other diagnostics have been used to obtain an approximate area of the fault. See Section MEMDBG

The first three tests, MEM256, MAP256 and DMA256, are run in the same way. To run MEM256, for example, type

MEM256 [ret]

with the CMI diagnostics disk in the left hand drive.

The standard command interpreter is used so the LIST and R commands, and P and N options are available as described in Chain file diagnostics. Tests are run by typing

<test name>[,<option1>,<option2>...,<optionN>] [ret]

To understand the information below, three definitions are required:

Logical or **Processor Space:** is the numerical range of addresses actually put out on the processor address bus. The logical address may also come from a DMA device such as the Floppy or Hard Disk controllers, MIDI card or Waveform Processor.

Physical Space: is the area of RAM which is actually accessed in response to the logical address.

Mapping: is the translation of any given logical address to any physical address. The Q256 does this on 2K "pages". Data which is contiguous within a single page of logical space will be contiguous within a page of physical memory. However all the pages that are contiguous within logical space can be arbitrarily shuffled in physical memory without the processor being aware of this when running programs etc. Also, an area of physical memory can appear in zero, one, two or more different places in the logical space.

Common Initialization, Options and Test Procedures

The three test programs share common initialization and option selection routines, and use common move and mapping routines to test the area of memory occupied by the operating system and test program.

Initialization: When the test program is first entered, the PIA setup of the graphics card is read and saved, for later restoration after tests which change this setup.
When the user types a command, one or all of a group of tests with the same name are executed sequentially. Before each group is run, the error counter is zeroed. Before each individual test is executed, a standard test mapping is set up in map 28, then all system states are switched to map 28. The standard test mapping is as follows:

Processor Space	Physical Memory
$0000 - $7FFF	Blocks 0.1 (Block 0 includes QDOS and test prog)
$8000 - $FFFF	Deselected

At the end of each test or during an abort caused by errors, the graphics card PIA setup is restored to the state which was saved on program entry. Then a check for any parity errors which occured during testing is performed, and a message printed if an error is found. The operating system and test program are moved back to the bottom of physical memory if necessary, and QDOS mapping reinitialized. Lastly the disk driver routines are reloaded to the top of processor RAM space from the floppy disk controller ROM and hard disk controller ROM if installed. The reloading of disk drivers is necessary because they are not part of the protected operating system and may have been overwritten by the memory tests.

Options: A number of options which apply to all tests can be set using the SEL command, rather than having to specify them each time a command is typed. To obtain a display of current option settings just type

SEL [ret]

To change an option, type

SEL,<option>=n [ret] where n is a number.

The options available are -

	Range	Default
C - Card select.	0-15	0
B - Block(s) select.	0-15	0-15.
E - Error limit.	0-127	0
D - Display	0-1	0
V - Video ouput	0-1	1
L - Line printer output	0-1	0

The Card select option allows multiple boards to be tested. This is useful in testing a board which is not working sufficiently to be able to load the operating system and test diagnostics. The default of zero tests the system card. Note that although the test can handle up to 16 cards, the standard CMI motherboard has only two Q256 slots.

Each card contains 256K bytes of memory consisting of four columns of 64K chips. The memory management hardware divides the 256K physical space into 128 pages of 2K each, but the bulk memory tests deal with sixteen blocks of 16K each (each 16K block therefore consists of 8 2K pages). The B option allows selection of which blocks are to be tested. The default option tests all blocks, from block F (15) downwards, which means the operating system block (Block 0) gets tested last.

The error limit determines how many errors can be logged before the test aborts.

The display option enables the printing of each card and block number as it is tested. Since several of the tests take a fairly long time to execute, particularly in a multi-card test rig, this option reassures the operator that the test is still running and has not crashed due to errors in the system memory.

The video option allows the above display to be printed on the system video console, while the L option prints it on a line printer if this is connected to the back panel of the CMI.

Error Messages

For most tests, if an error occurs a common error message routine is called which indicates where in memory the error occurred, the data which was expected, and the data which was actually read. It then reads the error location again. If on the second read the data is correct, the error is indicated as "SOFT". If it is still wrong and the same as it was the first time, it is indicated as "HARD". If the second read is wrong but different from the first read, the error is "RANDOM".

Testing Physical Block 0

The operating system (OS) normally resides in the bottom 16K of both logical and physical space, within processor addresses 0 to $3FFF. All testing takes place in the processor address range $4000 to $7FFF. When Block 0 is to be tested, the OS must be moved to a different block. The new block is first mapped into the address range $4000 to $7FFF then Block 0, residing in 0 to $3FFF is copied up to the new block. This moves QDOS, the test program, and Processor 2's stack into the new block. Processor 1's stack is in separate RAM on the Q133 and so is not affected by the Q256 diagnostics. At this point two copies of the OS exist, with P2 actually executing the one in Block 0. The new block is mapped into the address range 0-$3FFF so that now P2 is executing the new copy. Block 0 is then free to be mapped by the normal mapping routine into the $4000-$7FFF address range for testing. When the test on Block 0 is complete, the OS is again copied into $4000-$7FFF, then Block 0 is mapped back into 0-$3FFF.

MEM256
This section describes MEM256 V1.5.

MEM256 performs the bulk memory data and addressing tests, refresh and parity system tests.

Test name: 29
No. tests: 8
Purpose: Checks both processors' abilities to write rapidly varying data throughout memory.

In each test, the SELected blocks are mapped one at a time into $4000 to $7FFF and a semi-random data sequence written by either or both processors in an order as specified below. Each processor then checks its own data.

Test no.1: *P1 ODD UP, P2 EVEN UP*
P1 writes to all odd locations starting from $4001 and proceeds upwards, while P2 writes to even locations starting from $4000 and proceeds upwards.

Test no.2: *P1 ODD DOWN, P2 EVEN UP*
P1 writes to all odd locations starting from $7FFF and proceeds downwards, while P2 writes to even locations starting from $4000 and proceeds upwards.

Test no.3: *P1 ODD DOWN, P2 EVEN DOWN*
P1 writes to all odd locations starting from $7FFF and proceeds upwards, while P2 writes to even locations starting from $7FFE and proceeds downwards.

Test no.4: *P1 ODD UP, P2 EVEN DOWN*
P1 writes to all odd locations starting from $4001 and proceeds upwards, while P2 writes to even locations starting from $7FFE and proceeds downwards.

Test no.5: *P1 ODD, P2 EVEN*
P1 writes to all odd locations starting at $4001 then $7FFF and proceeds by writing alternately to bottom and top of memory working in towards the middle. P2 writes to even locations starting from $4000 and $7FFE and proceeds in a similar fashion towards the middle. The verify phase starts from the middle and works outwards.

Test no.6: *P1 EVEN, P2 ODD*
P1 writes to all even locations starting at $4000 then $7FFE and proceeds by writing alternately to bottom and top of memory, working in towards the middle. P2 writes to odd locations starting from $4001 and $7FFF and proceeds in a similar fashion towards the middle. The verify phase starts from the middle and works outwards.

Test no.7: *P1 ALL*
P1 writes to all locations starting at $4000 then $7FFF and proceeds by writing alternately to bottom and top of memory working in towards the middle. The verify phase starts from the middle and works outwards. P2 executes a dummy routine.

Test no.8: *P2 ALL*
P2 writes to all locations starting at $4000 then $7FFF and proceeds by writing alternately to bottom and top of memory working in towards the middle. The verify phase starts from the middle and works outwards. P1 executes a dummy routine.

Test name: WA
No. tests: 2
Purpose: Checks for addressing errors.

Test No.1: *WALKING ADDRESS P1*
Test No.2: *WALKING ADDRESS P2*
The test block is first filled with random data. Then each even/odd pair of bytes is written with the address of the even byte (e.g. $4000 is written to $4000,$4001). Each byte of data is verified immediately after it is written, and when the block has been filled, all data is verified again in read-only pass.
In test No. 1, P1 performs the test, while P2 executes a dummy, and vice-versa for test No. 2.

Test name: REF
No. tests: 4
Purpose: Checks that dynamic memory refresh is working.

Options:	T=n	Time option Range 1-100, default 5
	O=n	Operating system block Range 0-3, default 0

Test No.1 *REFRESH*
P2 fills all the test blocks except the OS block with $50 plus the block number, then executes a delay loop for T seconds. It then checks the data in each block. If refresh is not working, the default delay of 5 seconds is ample for the memory contents to largely decay away. P1 executes the same delay loop but does no testing.

Test No.2 *REFRESH WITH TAS, P2 ONLY*
P2 fills all the test blocks except the OS block with $50 plus the block number, then executes a delay loop for T seconds, exercising the CPU card indivisible instruction hardware while it waits. It then checks the data in each block. P1 executes the dummy wait loop and does no testing.

Test No.3 *REFRESH WITH TAS. P1 ONLY*
This is the same as test no.2, with the processors swapped.

Test No.4 *REFRESH WITH TAS. BOTH CPUS*
P2 fills all the test blocks except the OS block with $50 plus the block number. P1 and P2 then both execute a delay loop for T seconds, exercising the CPU card indivisible instruction hardware while they wait. P2 then checks the data in each block.

Test Name: PARITY
No. tests: 2
Purpose: Verifies operation of the parity checking system.

Options: D=n Data used in test
Range 0-255, defaults 170, 168 respectively.

Test No.1: *PARITY SYSTEM - EVEN*
Test No.2: *PARITY SYSTEM - ODD*
These tests use the special parity error generation hardware on the Q256 card to predictably generate a parity error. Test no. 1 uses data with even parity (170=$AA), and test no. 2 uses data with odd parity (168=$A8).
Only P2 runs the test. Initially it is in the A, or system state and map 28 is selected. Map 29 is configured with the same standard test mapping (Common Initialization, Options and test Procedures) then P2 B state is allocated to map 29 with the Parity Error Generate (PERGEN) bit set. Then on each test block, a quick fill (every 128th byte) of the test data as indicated by the D option is performed, followed by a special quick verify with parity error generation. For each verify, the parity register of the current test card is first read to check for any prior errors which should not be there yet. Then the CPU fuse register is set up to switch to the B state, just as the test data byte is read. Since the byte was written in the A state with PERGEN not set (as is normal), and is read back in the B state with PERGEN set, a parity "error" should be generated at the moment of reading. We then switch back to the A state and interrogate the current test card's parity status register to check the presence of the parity error flag, the correct physical 64K rank number, and correct upper 5 bits of the processor address at which the "error" occured. The register is then read again to check that the first status read automatically cleared the error.

MAP256
This section describes MAP256 V1.6

Test name: MAP
No. tests: 15
Purpose: Tests the mapping functions of the Q256.

Test no. 1 *PAGE UNIQUENESS - P2*
Test no. 2 *PAGE UNIQUENESS - P1*

Options: O=n1,n2...n4 OSBLK: Operating system block
Range 0-3, default 0-3
(4 options)

These tests check that each 2K physical page (there are 128 on each Q256) can be accessed uniquely. The general approach is to write each page with its own page number, and then verify them all to ensure no page overwrote another. Getting around the operating system complicates things slightly.
The test is actually run four times. According to the default OSBLK option it is first run with the OS in Block 0 (no moving required). Block 0 occupies pages 0 to 7 so pages 8 to 127 are mapped into $4000 to $7FFF in groups of 8 and "quick-filled" (every 128th byte) with the page number. Pages 8 to 127 are then quick checked to still contain their page numbers. Then the OS is moved to the second OSBLK option, Block 1, which occupies pages 8 to 15, so pages 0-7 and 16-127 may be tested with filling and verifying page numbers. Similarly the OS is moved to Block 2 while testing pages 0-15 and 24-127 and finally the OS is moved to Block 3 while testing pages 0-23 and 32-127. Which blocks are used for the OS and in what order may be changed using the O option.
Processor 2 executes test no.1, and P1 runs test no. 2; the procedure is the same for both.

Test no. 3 *P2 MAPPING, P1 TAS*
Test no. 4 *P1 MAPPING, P2 TAS*
This test ensures that the indivisible instruction hardware on the CPU card can be rapidly accessed simultaneously with mapping operations on the Q256. The MAPPING processor (P2 in test 3, P1 in test 4) follows the same procedure as in the page uniqueness tests 1 and 2. Again the test is run four times with the OS in one block while all other physical pages are filled and verified with page numbers. Meanwhile, the other processor loops around quickly doing two Test-And-Set instructions followed by two Clear instructions on a flag byte in memory. The flag byte is not used for any other purpose. The TAS processor continues testing and doing IO services until another flag is set by the MAPPING processor to indicate it has finished.

Test No. 5 *PROCESSOR UNIQUENESS*

The capability of the two processors to be mapped independently is tested by selecting different blocks in 2 different maps and running tests with different data on both processors simultaneously. The test is run once for each card under test. P1 is left as initialized in map 28, and Block 1 of the test card is mapped into $4000-$7FFF. P2 is switched to map 29 and Block 2 of the test card is mapped into P2's $4000-$7FFF. P1 then fills its test area with $AA while P2 fills its area with $55. The two processors then verify their own areas to contain the correct data.

Test No. 6 *FAST MAPPING - P2*

Test No. 7 *FAST MAPPING - P1*

These tests ensure that rapid writing to the mapram does not interfere with normal memory accesses. One processor loops over all test blocks (as selected by the SEL command) writing to all locations starting at $4000 then $7FFF and proceeding by writing alternately semi-random data to bottom and top of memory working in towards the middle. The verify phase starts from the middle and works outwards. (This is the same as the ALL tests in MEM256, N=7,8).

Meanwhile the other processor switches to map 29 and rapidly maps blocks in and out of $4000-$7FFF. All blocks except the current OS block and the block being tested by the first processor are mapped into $4000-$7FFF and quick-filled (every 128th byte) with zeros. After the last block, a flag is set to tell the first processor to stop testing.

In test no. 6, P2 does the fast mapping while P1 tests memory, and vice-versa for test no. 7.

Test No. 8 *SELECT ALL MAPS - P1*

Test No. 9 *SELECT ALL MAPS - P2*

32 different mappings may be set up in the Q256 mapram. Each map from 0 to 31 is initialised with Block 0 containing the OS mapped into 0-$3FFF and a different physical page mapped into logical page 8. (Physical page 8 is used in map 0, page 9 in map 1, ... page 39 in map 31). After each initialization the processor switches to that map, and fills logical page 8 with the map number. We then start again at map 0 and verify that its logical page 0 contains zero, and repeat through to map 31.

P1 runs the test in no. 8, P2 runs no. 9.

Test No.10 *A/B SELECT - P2*

Test No.11 *A/B SELECT - P1*

The Q256 allows for different mappings to be selected depending on whether the processor is in the A (System) or B (User) state. Map 29 is initialised with the standard test mapping and the B state is allocated map 29. Then the following procedure is repeated on each test card: starting in A state, Block 1 of the test card is mapped for all states into $4000-$7FFF (call this area K1) and filled with $AA. Block 2 is then mapped for all states into K1 and filled with $BB. Next Block 1 is mapped to K1 for A states only, and Block 2 is mapped to K1 for B states only. Still in the A state, K1 is verified to contain all $AAs.

Then the processor switches to the B state and verifies K1 to contain all $BBs. Thus far it is proven possible to read from different areas of physical memory in the two processor states.
The second stage of the test switches back to the A state and fills K1 with $11. Then the processor switches to the B state again and fills K1 with $22. Still in the B state, Block 1 is mapped into K1 for all states and checked to still contain $11 then Block 2 is mapped into K1 for all states and checked to contain $22. This proves that data written while in the A or B state can be directed to different areas of memory. The processor switches back to the A state for the next card to be tested or to exit the test.
P2 runs test no. 10; P1 runs test no. 11.

Test No.12 *PERIPHERAL DISABLE - P1*
Test No.13 *PERIPHERAL DISABLE - P2*

These tests check the peripheral enable output from card 0 which allows the peripherals (i.e. everything on the bus except the RAM card itself) which usually reside above $E000 in the processors' logical address space, to be disabled and replaced by extra RAM. The Q256 mapping system is itself a peripheral - the mapram lives from $F000-$F7FF (one logical page) so this test works by trying to change the mapping when peripherals are disabled.
Map 29 is initialised with the standard mapping and the B state is allocated map 29 with peripherals disabled. Then for each test card the following procedure is followed:
Still in the A state (and map 28), physical pages 8 and 9 are mapped to logical pages 8 and 9 respectively. Page 8 is filled with $AA and Page 9 is filled with $55. Then for map 29, physical page 9 is mapped to both logical page 9 and logical page 30 i.e. to the same logical area as the mapram. From the B state, reading the mapram area would now get all $55s.
We switch to the B state and attempt to map logical page 9 to physical page 8 in map 29 (the B map). With peripherals disabled, this should not work, and logical page 9 is verified to still contain $55.

If all is well so far, the Peripheral Enable probably works, so we attempt to completely clobber the mapram by filling it with $AA from the B state. Since peripherals are disabled in the B state, this should merely write to physical page 9 without changing any mapping. Logical page 30 is verified to contain $AA, then we switch back to the A state and verify logical pages 8 and 9 as both containing $AA. (Since peripherals are disabled, the last state switch can not be achieved by writing to the processor card control latch - a dummy software interrupt is called instead. The A state is automatically selected on interrupts.)
P1 runs test no. 12; P2 runs test no. 13.

Note - Only Card 0 is effectively tested by this test because it is the only card whose Peripheral Enable output pin is connected on the motherboard to its own and all other RAM card's PENB inputs.

Note - also that if an error occurs while testing in the B state, the A state must be reselected in order to print the error message. The standard error message routine re-reads the incorrect location to ascertain what kind of error has ocurred, and since the mapping has changed it will no longer be reading the correct physical location. Thus errors which are in fact "Hard" will be indicated as "Soft", or possibly "Random".

Test No. 14 *GRAPHICS RAM DISABLE - P1*
Test No. 15 *GRAPHICS RAM DISABLE - P2*
Option: R=n
Range 0-1, default 0

Checks VRAMEN output from the memory management system which allows the 16K of dedicated RAM on the Q219 graphics card to be mapped in or out of either processor's address space.
The video graphics RAM (VRAM), if enabled, occupies the address range $8000 to $BFFF. The contents of VRAM are copied down to $4000 to $7FFF to be saved while testing takes place. Then VRAM is checked to be working by filling with $AA and verifying (the $AA pattern appears on the console screen). Map 29 is initialised with the standard mapping and the processor switches to map 29 with graphics access disabled. It attempts to clear VRAM then reselects map 28 with graphics enabled to verify $AA is still there. Finally the saved contents of the screen are copied back.
P1 runs test no. 14; P2 runs test no. 15. Normally only one processor has access to the VRAM and this is selected through the PIA control on the Q219 at system start-up and by some application programs. Both tests 14 and 15 can run because the PIA set-up is saved on entry to MAP256 and restored at the end of each test. The tests also run on the old graphics cards Q218 and Q045 but in the latter case, which processor has graphics card access is hardware selected, so one of the tests will exit without testing.
Note - Only Card 0 is effectively tested by this test because it is the only card whose VRAM Enable output pin is connected on the motherboard to the graphics card.

DMA256
This section describes DMA256 V1.6

Test Name: DMA
No. tests: 2

Test no.1 *P2 DMA MAP SWITCHING*
Test no.2 *P2 DMA/P1 UNIQUENESS*

DMA256 tests the automatic selection of special DMA mappings via the DMA claim lines which are input to the Q256 memory cards from all devices on the bus which use DMA (Floppy Disk Controller, Hard Disk Controller, General Interface card).

P2 initializes map 29 with Block 2 of the test card mapped into K1 (logical area $4000 to $7FFF) as a 16K DMA buffer, and switches into map 29 to fill the buffer with its own page numbers. Then P2 switches back to the standard map 28 which has Block 1 mapped into K1 and fills it with a semi-random sequence. A disk operation is then initiated which writes the DMA buffer out to disk and into a file called DMAFILE.TF. Map 29 is selected again and the DMA buffer cleared. P2 switches back to map 28 and starts a second DMA operation, this time reading the buffer in from disk. K1 in map 28 is checked first to be still containing the random sequence, then map 29 is selected to verify the page numbers read in from disk. The procedure is repeated for all test cards.

In test no. 1, only P2 runs the test. In test no. 2, P2 runs the same test while P1 continually random fills and checks its own separate block (Block 4 of the test card) in the same logical test area using map 27.

The test file DMAFILE.TF is not deleted at the end of the test so that in the event of errors, the file may be examined using the QDOS command DUMP (this is not the same as the Channel Card diagnostic DUMP command!). It is not required to exist before the DMA test runs, so the user may delete the file at any time.

MEMDBG

The diagnostic programs MEM256, MAP256 and DMA256 provide the comprehensive tests which thoroughly check the Q256 256K Ram card operation. However the size and complexity of these diagnostics mean that they are not very easy to use for chip-level debugging of known faulty boards. MEMDBG is a small and simple program which exercises specific sections of the Q256 circuitry in very tight loops and thus generates the stable CRO traces needed to find circuit malfunctions. The absence of error checking means that the processor does not end up spending most of its time printing error messages. It is only appropriate to use MEMDBG once MEM256 or the other diagnostics have been used to obtain an approximate area of the fault.

Use of MEMDBG

To start MEMDBG, type

MEMDBG [ret]

with a diagnostics disk containing MEMDBG.CM in the left hand drive. When the program is first entered, it calculates its own checksum in case memory faults have prevented a successful load. If the checksum is wrong, the program exits back to QDOS. This check can be overidden by typing

MEMDBG;-C [ret]

to start the program.

The user is presented with a menu of tests indicating which parts of circuitry are exercised by each test. Simply type the number of the required test then answer the prompts for data such as which processor is to run the test, what data is to be used, what 16K memory block is to be written/read etc.

Once all data required for the test is obtained the test loop is entered immediately. The only thing left to do then is watch the signals of interest behave as the test loop requires. In general it is best to connect the CRO before starting the program so that accidental shorts don't crash the system before you get to see anything. To keep the loops as tight as possible, there is no provision for terminating the test - just hit reset or console interrupt (NMI). The latter only works if P2 is running the test, and the front panel P2 NMI switch is enabled only. After a console interrupt the program may be re-entered from the monitor without reloading by typing

2000;G

A board which is unable to load the system and/or MEMDBG can only be debugged by optioning it as Card 1 and installing it along with a healthy board as Card 0. If the faulty board still crashes the system, disable the data driver using option W3 (see Q256 hardware documentation).

Adding Your Own Test Loops

The eight tests contained in the program should be sufficient to solve most problems on the Q256. However MEMDBG has been written such that debug technicians can easily add their own test loops tailor made to investigate particular curly problems if the need arises. To do this a disk containing the following files should be placed in the right hand drive:

MEMDBG.SA	Source file
MEMDBG.CF	Chain file for editing and assembling
QEQU.SA	QASAR system equates
MAC256.SA	Q256 mapping macros

A system disk containing the following files should be booted in the left hand drive:

RASM09.CM	6809 assembler
EDIT.CM	Cyword editor
IO.CM or QIOPACK.CM	QASAR IOpack required for editor.

A 6809 assembler manual will also be required.

The procedure for adding a new loop is contained in the chain file. Just type

CHAIN MEMDBG:1 [ret]

to start the process. The chain calls the editor followed by the assembler to generate a new MEMDBG.CM on drive 1. Options in the chain include:

C - generate an assembly listing in the console screen
P - print assembly listing on the line printer
L - save assembly listing as the file MEMDBG.AL:1

For example, type

CHAIN MEMDBG:1;C [ret] to list to the console.

The program consists of three sections: A test executive, the menu, and the collection of test loops. Only the menu needs to be modified to include the new loop, which should be typed into the source file after the other loops. Refer to the fully commented source file for further information on its structure.
After adding another test loop the version number and "last modified" message should be updated to distinguish the new program from old versions.
Once the program is modified, the checksum will no longer be correct. Use the -C option to run the program and get the checksum error message. The formula for calculating the new checksum is:-

new CHKSM = complement(old CHKSM - error CHKSM reported) + 1

Run the chain again and modify the CHKSM byte to the new value.