The ATM Camera V2 (AVA200)

Ian Pratt and Paul Barham

15th March 1994

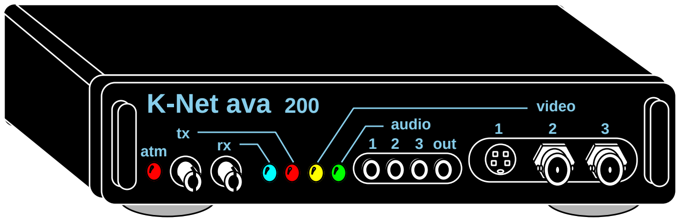

Figure 1: AVA200 Physical Appearance.

1 Overview

The ATM Camera V2 is a high performance ATM peripheral for the digitization of live audio and video directly into ATM cell streams. The device is highly pipelined and thus exhibits a very low latency.

Two variants of the ATM Camera V2 are expected to exist. One will be the standalone AVA200, and the other will be modified for direct connection to the Fairisle switch fabric for use in Desk Area Network applications.

The AVA200 version is designed for desktop use and is housed in an attractive custom designed box shown in Figure 1. The unit has 3 physical video channels and 3 audio channels. One of the video connectors is usually an SVHS connector [1] and the other two are 75Ω BNC connectors, although any combination of these inputs is possible. The SVHS connector has separate luminance and chrominance inputs giving much better picture quality and should be used where a high quality video source is available.

The audio connectors are 3.5mm stereo jacks with the left channel connected to the tip as per most audio standards. All audio inputs are nominally line level [2], although Audio input 1 may be configured to accept microphone level signals [3].

The AVA200 network interface is ATM Forum standard 100Mbps or 140Mbps TAXI over optical fibre, whereas the DAN variant has the standard 96-way Fairisle backplane connector allowing direct connection to the DAN. The simplification of the ATM interface makes the DAN version cheaper to produce.

The unit has 5 LEDs on the front panel. From the left these are:

- The **bicolour** ATM LED is **red** when the fibre Rx connection is not receiving photons, and **green** when there is light.

[1] Otherwise known as 4-pin mini-DIN connectors.

[2] Approx. 2V RMS

[3] This is achieved by enabling an extra 20dB gain stage within the audio codec.

- The **blue** RX LED is on when the microcontroller is in the ATM Rx interrupt handler.
- The **red** TX LED is on when the transmission Xilinx is sending a cell.
- The **yellow** VIDEO LED is on when the DMA Xilinx is capturing a frame into the video FIFOs.
- The **green** AUDIO LED is on when the DMA Xilinx is capturing audio from the codec.

The rear panel contains an RS232C serial line over which diagnostics are output. A simple debugging monitor may also be accessed via this interface. The use of this interface is described in Section 6. The power switch is also located on the rear panel. The other connectors are an 8-pin mini-DIN for the external power supply, and an auxiliary connector which provides power and an I^2C bus for controlling peripherals such as camera pan/tilt mounts [4].

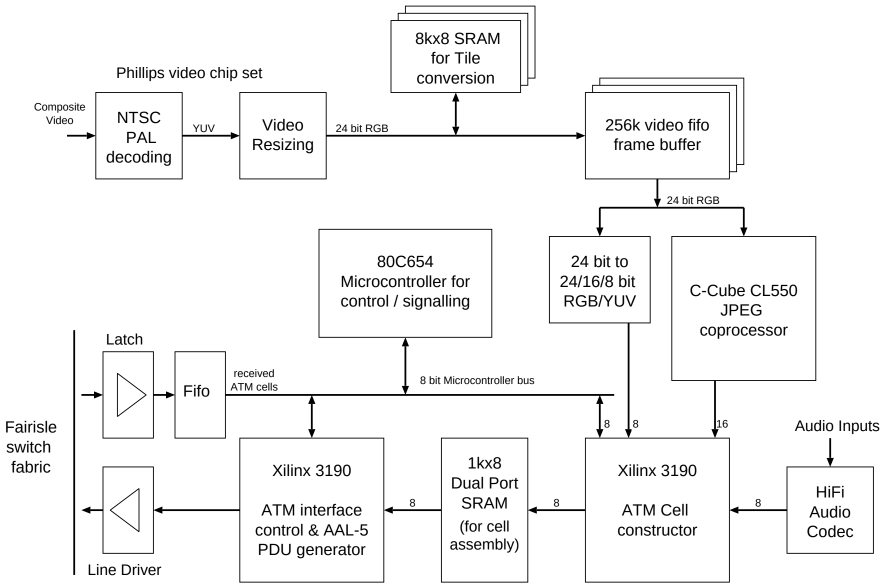

Figure 2: ATM Camera V2 Major Datapaths (DAN variant).

2 Hardware Description

The main datapaths within the ATM Camera V2 are shown in Figure 2.

Analogue video from the connectors on the front panel is digitised immediately, either as separate luminance and chrominance channels (for the SVHS case) or as a single PAL/NTSC composite signal. The digital sample stream is decoded by the Philips SAA7191 Digital Multistandard Decoder (Square Pixel) to produce a stream of 16-bit YUV pixels and all necessary horizontal and vertical sync signals. The decoder will automatically recognize PAL, NTSC or SECAM signals and adjust accordingly.

The output of the decoder passes into the SAA7186 Digital Video Scaler chip which can take a specified rectangle of the input picture and scale it to any smaller size. In addition, it can perform a number of colour-space conversion functions. Typically the output from the scaler chip is in the RGB colour-space since this it what most frame-buffers require. Both the Decoder and Scaler are controlled by the Microcontroller via an I^2C bus.

The pixels from the scaler pass through an 24-bit wide SRAM large enough to hold 8 scan-lines of pixels. They are written into the SRAM in scan-line format and read out as 8x8 pixel tiles. At

[4]The infamous AVC-200.

this point the data is written into video FIFOs large enough to store an entire frame. This allows the capture of the pixel data to proceed completely decoupled from its transmission across the ATM network interface.

The data from the VFIFOs may be transmitted as one of a number of uncompressed pixel formats by simple multiplexing of the bits, or may be passed through a JPEG compression engine (C-Cube CL550) to produce a variable-bit-rate stream.

A HI-FI audio Codec provides audio samples at a number of rates, in selctable formats up to 48kHz stereo 16 bit samples. The codec has 3 inputs and an output currently used for monitoring the capture channels.

The Microcontroller can also contruct cells for transmission over the ATM interface using registers provided within this Xilinx. This allows the MPU to transmit synchronization streams etc.

The bytes which result from all of these operations are marshalled into AAL5 PDUs by a Xilinx 3190 FPGA. The cell assembly takes place in a dual-ported SRAM.

Another Xilinx on the other side of the DPSRAM manages the ATM interface and is responsible for setting up the length, CRC32 and the header flag fields of each cell transmitted. This Xilinx supports four independent transmission channels:

- Raw video
- JPEG compressed video
- Audio
- Microcontroller IO

Each channel is capable of multicasting to four destinations, although this feature is not currently used.

Of the four channels, the MPU has priority to avoid the situation where control information is unable to get through due to the high bandwidth requirement of the ATM video streams.

On the AVA200 variant this Xilinx controls the two TAXI chips responsible for the ATM transmission interface. On the DAN variant the Xilinx presents the cell data directly to the switch fabric via two latches.

ATM Rx is also supported by the transmission Xilinx which provides a method for the MPU to read the 9-bit wide Rx FIFO. The 9th bit is used to tag the start of each cell. The microcontroller is interrupted when this FIFO is non-empty.

3 Example System

The system shown in Figure 3 illustrates the expected design of multiple client video and audio applications across a LAN or WAN. The design of the AVA200 firmware is intended to support use of this kind.

3.1 Description

The diagram shows an AVA200 connected to a port on a switch. A manager process is running on a workstation connected to the same switch, although all that is necessary is that the manager is in close proximity to the AVA200, providing a semi-reliable communications path. This means that simple protocols may be used to talk to the AVA200, and the guaranteed low latency allows finer grained scheduling of the hardware.

Client processes are run on workstations which are also connected to switches. These may be remote, possibly across a WAN causing problems of relatively high latency and cell loss for management operations. The client processes therefore communicate with the manager process using RPC (for example) to negotiate their video and audio requirements. The transport layer should be chosen bearing in mind reliability and latency constraints. The manager allocates

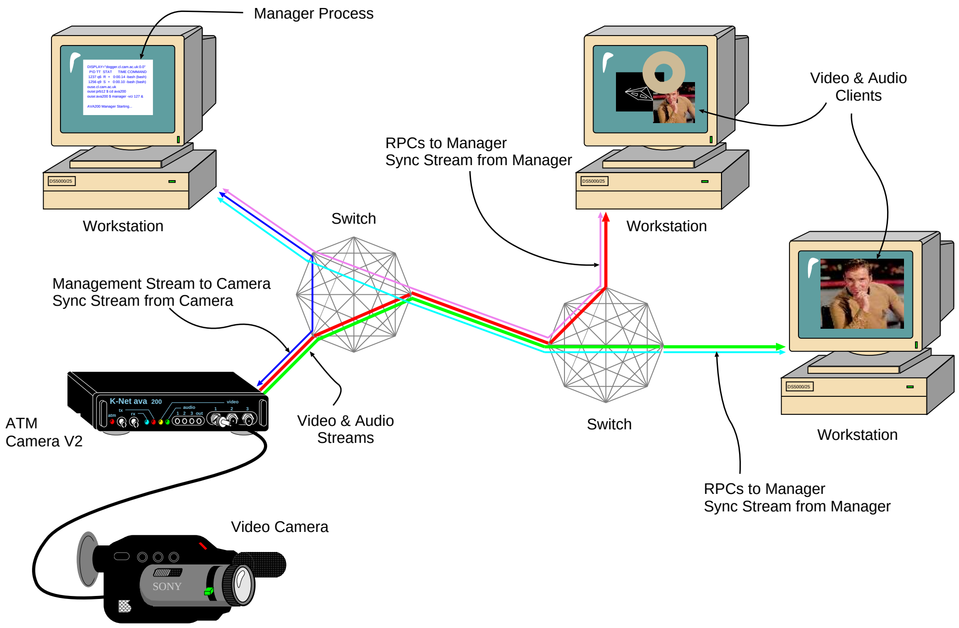

Figure 3: AVA200 Example System.

resources between clients and computes suitable schedules for the AVA200. These are sent to the AVA200 on the *management* VCI unsing the AVA200 control protocol.

Video and audio streams are sent directly from the AVA200 to the client processes. This removes as many shared resources as possible from the datapath, allowing QoS constraints to be achieved more easily. If the data streams had to pass through the manager this would produce a substantial bottleneck, as well as presenting a large resource management/scheduling problem.

Synchronization information for all audio and video channels is sent to the manager process where suitable sync streams for each client may be constructed. This simplifies the firmware required in the AVA200, and allows sync streams of any format to be produced. If the manager is in control of multiple AVA200s then in may produce composite sysnc streams etc. Since the sync streams are low bandwidth this presents no problem for the manager.

The client processes are required to send per-stream "KeepAlive" messages to the manager to inform the manager that they are still prepared to receive video and audio. These are forwarded to the AVA200 on the *management* VCI to minimise security problems. The keepalive messages are of the form "*send me up to frame number N*". In the situation where the clients are swamped by too much incoming audio and video and unable to transmit keepalives, the AVA200 runs out of credits on the appropriate streams and they stop automatically. A timeout on the *management* VCI detects the case where the management process is unable to communicate with the AVA200 and stops all streams.

4 AVA200 Firmware and Associated Control Protocol

Since the hardware only contains a single digital video chipset, and a single audio codec, only one physical input may be used at a time. When using "gen-locked"[5] video sources, it is possible to multiplex the digitising hardware on a per-frame basis. For audio this is clearly not sensible.

[5]PAL encoding follows an 8 frame sequence so resynchronisation between signals which are not at the same point within the sequence can be expensive.

The firmware therefore supports multiple concurrent video streams, [6] and a single outgoing audio stream.

For each video stream it is necessary to download a number of parameters to the unit. These parameters specify picture sizes and scaling, pixel formats, data rates, the physical video channel to use and the VPI/VCI for the outgoing video stream. This information is loaded into a "Video Bucket". Similarly, parameters for the audio streams including sample rate, format and physical audio input must be loaded into an "Audio Bucket".

A "Schedule" is then loaded into the AVA200 which specifies a sequence of video buckets. This schedule is executed in a round-robin fashion, grabbing a frame using the parameters specified in the relevant video bucket. Thre reserved video bucket index of zero indicates that the unit should idle for a frame. The schedule also identifies which audio bucket should be used.

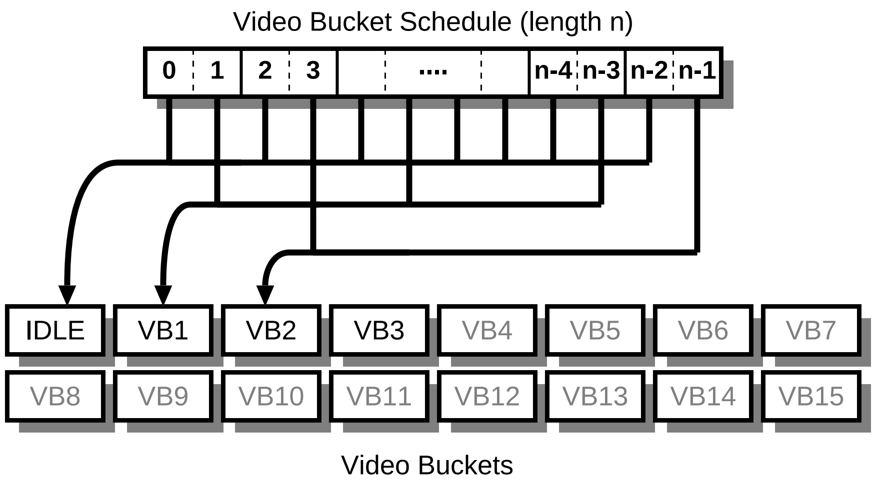

Figure 4: AVA200 Video Programming Paradigm.

These datastructures are shown diagrammatically in Figure 4. The figure illustrates two concurrent streams whose parameters are specified in Video Buckets VB1 and VB2. The first stream is being allocated 3 frames out of every n, and the second is receiving 2 frames out of every n. The remaining frames are being spent idling. Bucket VB3 has been loaded with the parameters for a third stream in preparation for an atomic change of schedule.

The audio paramters for each codec are selected from one of a number of Audio Buckets (see Figure 5). Since the AVA200 has only one codec, only the bucket referenced by channel A0 is examined. Audio buckets may also be loaded in advance (e.g AB1) in preparation for an atomic change of configuration.

The AVA200 uses AAL5 for all network I/O, although this may not be the case for the DAN implementation as the reliable nature of communications across the DAN remove the need for the added complexity of AAL5. Video formats used on the DAN may also take account of the unique properties of the Desk Area Network.

The AAL5 CRC32 field is not checked by the MPU on receive however, due to the complexity, so a simpler 16-bit checksum is used. The AAL5 CRC32 field is correctly generated for all transmitted PDUs however.

The receive side of the AVA200 control protocol consists entirely of single cell (52/53 byte) AAL5 PDUs. This is to remove the need for complicated reassembly functions on board the unit. The same format is used for both requests and acknowlegements.

[6]The current firmware and associated protocol support up to 15 video streams, limited by the four-bit Video Bucket Indices (VBIs) in the SCHEDULE cell.

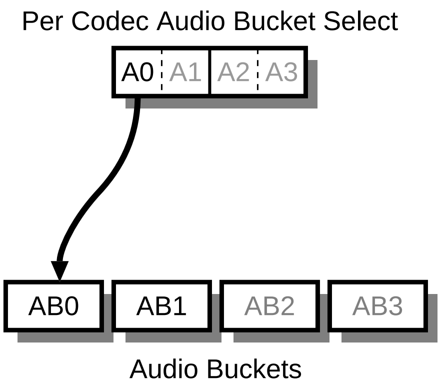

Figure 5: AVA200 Audio Programming Paradigm.

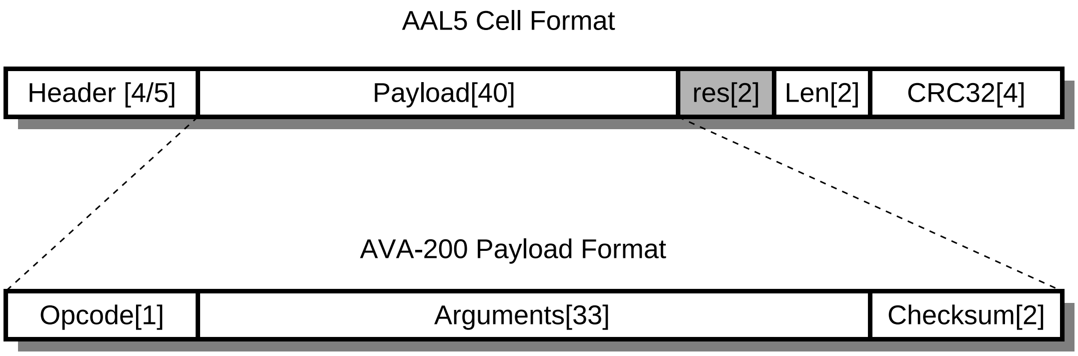

Figure 6: AVA200 Generic Protocol Cell.

4.1 Generic Control Cell Format

Within this AAL5 PDU are various fields which are common to all control cells. A generic AVA200 control cell is structured as shown in Figure 6. The offsets of the fields within a cell are summarised in Table 1. The various fields are described below:

- SEQUENCE - If these four octets match those in the previous cell then this cell is treated as a duplicate of the previous cell. Duplicates will be acknowleged without any further action.
- OPCODE field identifies the type of the control cell as shown in Table 2. The most significant bit of the opcode field is set if the cell is an acknowlegement cell.
- CHECKSUM contains two octets which represent a simple 16 bit checksum of the previous 38 octets. Although the entire block is protected by the AAL5 checksum, this is too expensive

Field	Offset
PROTO_SEQUENCE	0
PROTO_OPCODE	4
PROTO_USER_DATA	5
PROTO_CHECKSUM	38

Table 1: Generic Protocol Cell Field Offsets

Opcode	Value
PROTO_OPCODE_PING	0
PROTO_OPCODE_RESET	1
PROTO_OPCODE_SCHEDULE	2
PROTO_OPCODE_VIDEO	3
PROTO_OPCODE_AUDIO	4
PROTO_OPCODE_VIDEO_KEEPALIVE	5
PROTO_OPCODE_AUDIO_KEEPALIVE	6
PROTO_OPCODE_SYNC	7

Table 2: Protocol Cell Opcodes

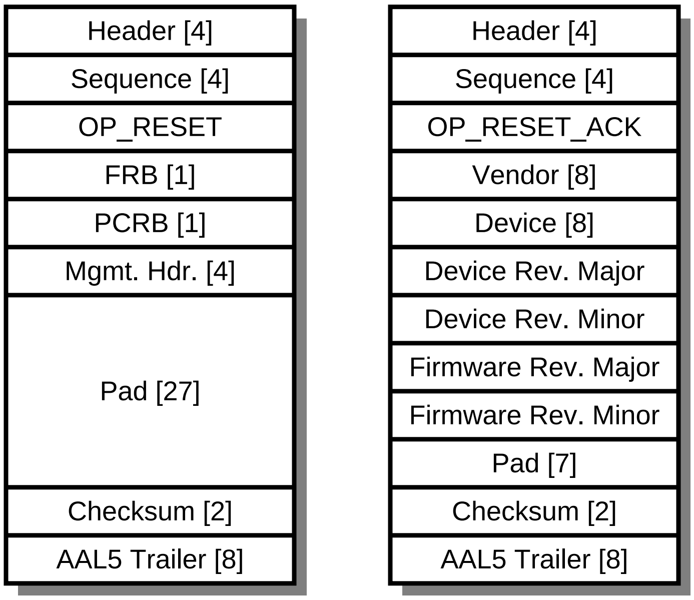

Figure 7: AVA200 Reset Cell and Acknowlegement.

to compute with the limited processing power available on the unit, and a simpler error detection mechanism was therefore required. The checksum consists of the sum modulo 2^{16} of the previous 38 octets XORed with 5555_{16}. The most significant octet is first.

4.2 Reset Cell Format

Sending a Reset cell to the AVA200 will cause all subsystems of the unit to be reinitialized. The management VPI/VCI for all future communications is specified in the cell. The acknowlegement for the cell contains version information for the hardware and firmware. The AVA200 Reset cell is structured as shown in Figure 7. The various fields in the Reset cell are described below:

- Frb - Fabric route byte (1 octet). This field is only examined if the AVA-200 is connected directly to a switch fabric.
- Pcrb - Port Controller route byte (1 octet). This field is only examined if the AVA-200 is connected directly to a switch fabric.
- Mgmt. Header - ATM Header for the management stream (4 octets).

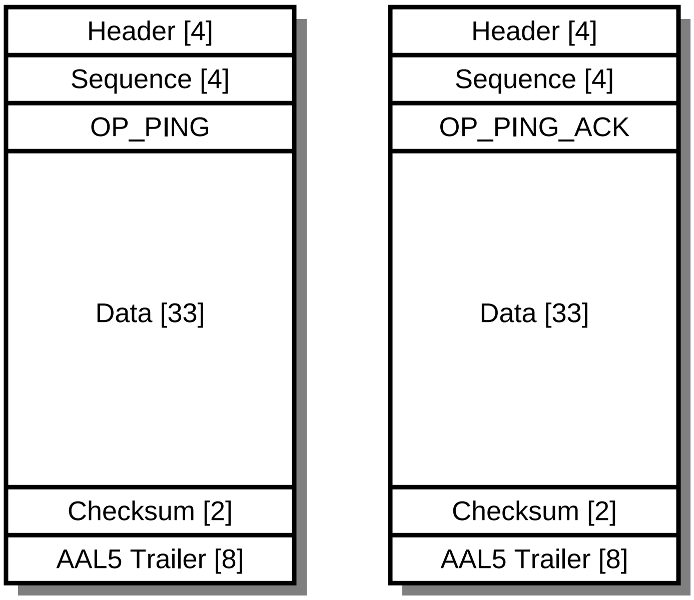

Figure 8: AVA200 Ping Cell and Acknowlegement.

The fields in the RESET acknowlegement are described below:

- VENDOR - Contains an ASCII string identifying the vendor.
- DEVICE - Contains an ASCII string identifying the device.
- DEVICE REV MAJOR and DEVICE REV MINOR identify the revision number of the unit.
- FIRMWARE REV MAJOR and FIRMWARE REV MINOR identify the revision number of firmware installed in the unit.

4.3 Ping Cell Format

Sending a PING cell to the AVA200 will cause the cell to be acknowleged without any further action. This is useful for checking the reachability and state of the unit. The acknowlegement for a ping request contains the same data as the request cell. The AVA200 PING cell is structured as shown in Figure 8.

4.4 Video Load Cell Format

Sending a VIDEO cell to the AVA200 enables the loading of per-stream configuration into one of the 16 available buckets. The AVA200 VIDEO cell is structured as shown in Figure 9. The offsets of the fields within the Video Bucket are summarised in Table 3. The INDEX field identifies which bucket is to be loaded. Only the least significant four bits are examined. The format of the configuration data to be loaded into the bucket is shown in Figure 10. The video data consists of:

- FRB - Fabric route byte. This field is only examined if the AVA-200 is connected directly to a switch fabric.
- PCRB - Port Controller route byte. This field is only examined if the AVA-200 is connected directly to a switch fabric.
- STREAM HEADER - ATM Header for the video stream.

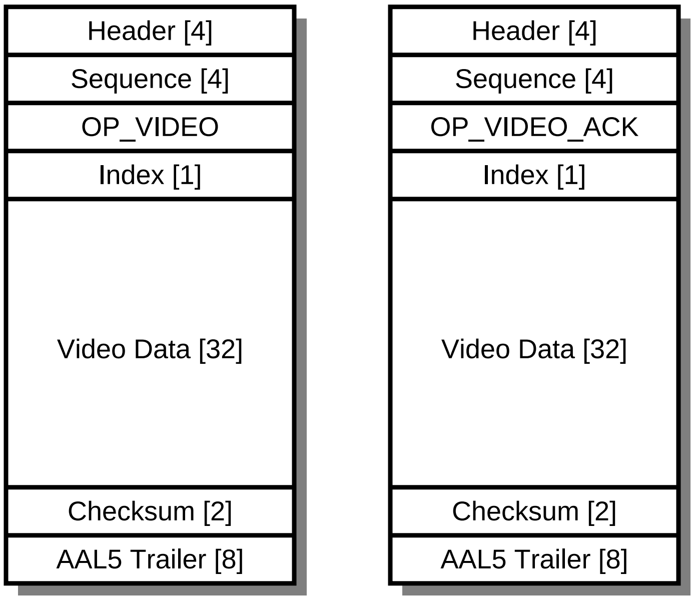

Figure 9: AVA200 Video Load Cell and Acknowlegement.

Field	Offset
VIDEO_FABROUTE	0
VIDEO_PCROUTE	1
VIDEO_HDR	2
VIDEO_XIREGS	6
VIDEO_CONTROL	6
VIDEO_RATE	7
VIDEO_XTILES	8
VIDEO_ZTILES_MSB	9
VIDEO_ZTILES_LSB	10
VIDEO_PACK	11
VIDEO_LENGTH_MSB	12
VIDEO_LENGTH_LSB	13
VIDEO_TIMEOUT	14
VIDEO_7186	15
VIDEO_7191	24
VIDEO_SEQUENCE	25
VIDEO_KEEPALIVE	29

Table 3: Video Bucket Data Offsets

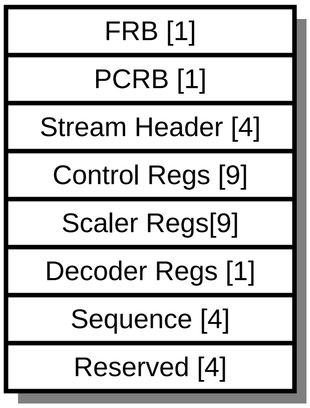

Figure 10: AVA200 Video Data.

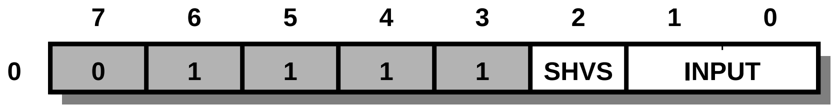

Figure 11: Decoder Registers.

- Control Regs - Register values for the control logic as shown in Figure 16.
- Scaler Regs - Register values for the Digital Video Scaler (Philips SAA7186). For more information see Figure 14
- Decoder Regs - Register values for the Digital Multistandard Decoder (Philips SAA7191). For more information see Figure 11
- Sequence - Contains the initial video frame sequence number. If the first octet of the sequence number is non-zero then the sequence number is loaded from the bucket number it specifies.
- Keepalive - Contains the initial number of video frames to transmit before keepalive messages are required. Some early units do not support the keepalive feature.

4.4.1 Decoder Registers

The fields in the Decoder registers have the following effects:

- Input selects one of the 3 inputs.
- SVHS should be set when an SVHS input is being used i.e. Separate luminance and chrominance information. Without this bit set, only the luminance information will used and the picture will appear monochrome.

4.4.2 Scaler Registers

The fields in the Scaler registers have the following effects:

- Field selects whether to grab odd fields only, even fields only or both as follows:

  00 Reserved.

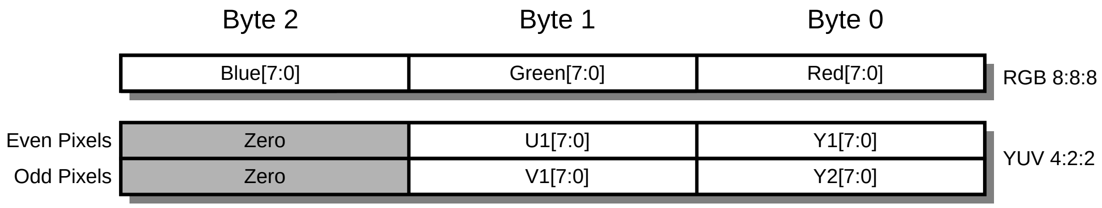

Figure 12: Pixel Formats output by the Scaler.

01 Both fields for non-interlaced storage.

10 Odd fields only for non-interlaced storage.

11 Even fields only for non-interlaced storage.

- FORMAT selects the format of the pixel data supplied by the front end from one of those shown in Figure 12. Note that the bus is 24 bits wide, and so for YUV 4:2:2 the U and V components alternate between odd and even pixels.

  01 16 bit YUV 4:2:2

  10 24 bit RGB 8:8:8

- XD sets the pixel width of the output from the scaler.
- XS sets the pixel width of the input to the scaler.
- XO sets the horizontal pixel offset of the input to the scaler.
- YD sets the pixel height of the output from the scaler.
- YS sets the pixel height of the input to the scaler.
- YO sets the vertical pixel offset of the input to the scaler.

The relationship of these parameters is illustrated in Figure 13. By setting these values appropriately an application is able to achieve a correct aspect ratio using a raw, single field, video stream. It is possible to scale the full image size down to a single tile size if desired.

4.4.3 Control Registers

The fields in the control registers have the following effects:

- MODE selects the packing of the pixel data into the output bytes as shown in Figure 15. The input bytes may be 24 bit RGB 8:8:8 with the red component in byte zero, or 16 bit YUV 4:2:2 with the Y component in byte zero.

  00 8:8:8 RGB or 16 bit YUV depending on output of the scaler.

  01 5:5:5 RGB

  10 3:3:2 RGB

  11 5:5:5 BGR

- COUNT selects the number of bytes transmitted per pixel.

  00 1 byte per pixel e.g. 3:3:2 RGB or 8 bit mono.

  01 2 bytes per pixel e.g. 5:5:5 RGB or 16 bit YUV.

  10 3 bytes per pixel e.g. 8:8:8 RGB.

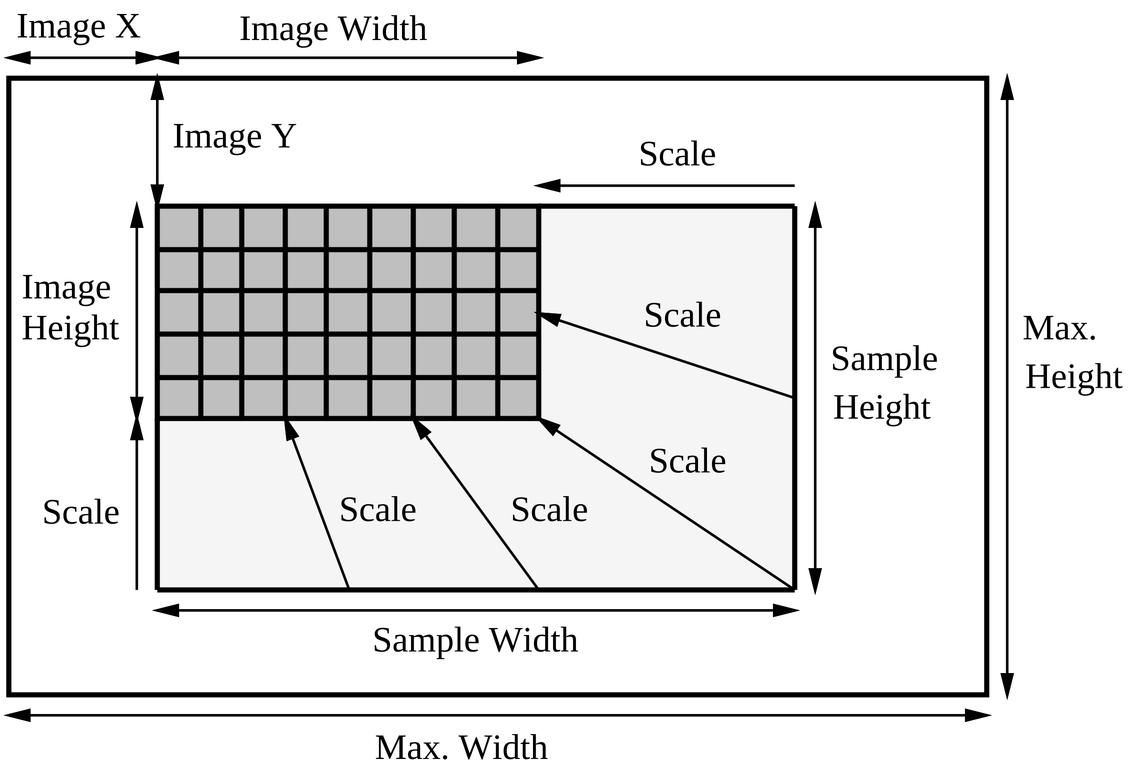

Figure 13: AVA Video Stream Sampling/Scaling

	7	6	5	4	3	2	1	0
0	0	FIELD		1	FORMAT			
1	XD[0-7]							
2	XS[0-7]							
3	XO[0-7]							
4	res			XO[8]	XS[8-9]		XD[8-9]	
5	YD[0-7]							
6	YS[0-7]							
7	YO[0-7]							
8	1	res		YO[8]	YS[8-9]		YD[8-9]	

Figure 14: Scaler Registers.

Field	Offset
AUDIO_FABROUTE	0
AUDIO_PCROUTE	1
AUDIO_HDR	2
AUDIO_LENGTH_MSB	6
AUDIO_LENGTH_LSB	7
AUDIO_1848	8
AUDIO_ENDIAN	17
AUDIO_SEQUENCE	24
AUDIO_KEEPALIVE	28

Table 4: Audio Bucket Data Offsets

11 4 bytes per pixel e.g. 8:8:8 RGB with a zero fourth byte.

- END reverses the order of the pixel bytes. e.g. 0123 → 3210. This can be used to adjust the order of Red, Green and Blue components to match the destination framestore.
- RATE controls the transmission rate for the pixel data. The small gap between transmission of each PDU is proportional to this value. The value must therefore vary with the size of PDU in order to maintain the same sustained transmission rate.
- XTILES sets the width of the frame in tiles.
- COUNT sets the number of PDUs per frame.
- PACK sets the number of 8x8 pixel tiles packed into each AAL5 PDU.
- LENGTH MSB sets the MSB of the AAL5 Length field in each tile.
- LENGTH LSB sets the LSB of the AAL5 Length field in each tile.
- TIMEOUT sets the timeout for grabbing each video frame in units of 4ms. This is used to prevent the unit from locking up when presented with a poor video signal.

4.5 Audio Load Cell Format

Sending an AUDIO cell to the AVA200 enables the loading of a new audio configuration. The data field of the AVA200 AUDIO cell is structured as shown in Figure 17. The offsets of the fields within the Audio Bucket are summarised in Table 4. The INDEX field identifies which audio bucket is to be loaded. The AUDIO DATA field consists of:

- FRB - Fabric route byte (1 octet). This field is only examined if the AVA-200 is connected directly to a switch fabric.
- PCRB - Port Controller route byte (1 octet). This field is only examined if the AVA-200 is connected directly to a switch fabric.
- STREAM HEADER - ATM Header for the audio stream (4 octets).
- LENGTH - sets the length field of the AAL5 PDU.
- CODEC REGS - Register values for the audio codec (Analog Devices AD1848) as shown in Figure 19 (16 octets).
- SEQUENCE - Contains the initial audio sequence number (4 octets).
- KEEPALIVE - Sets an initial number of audio PDUs to be transmitted.

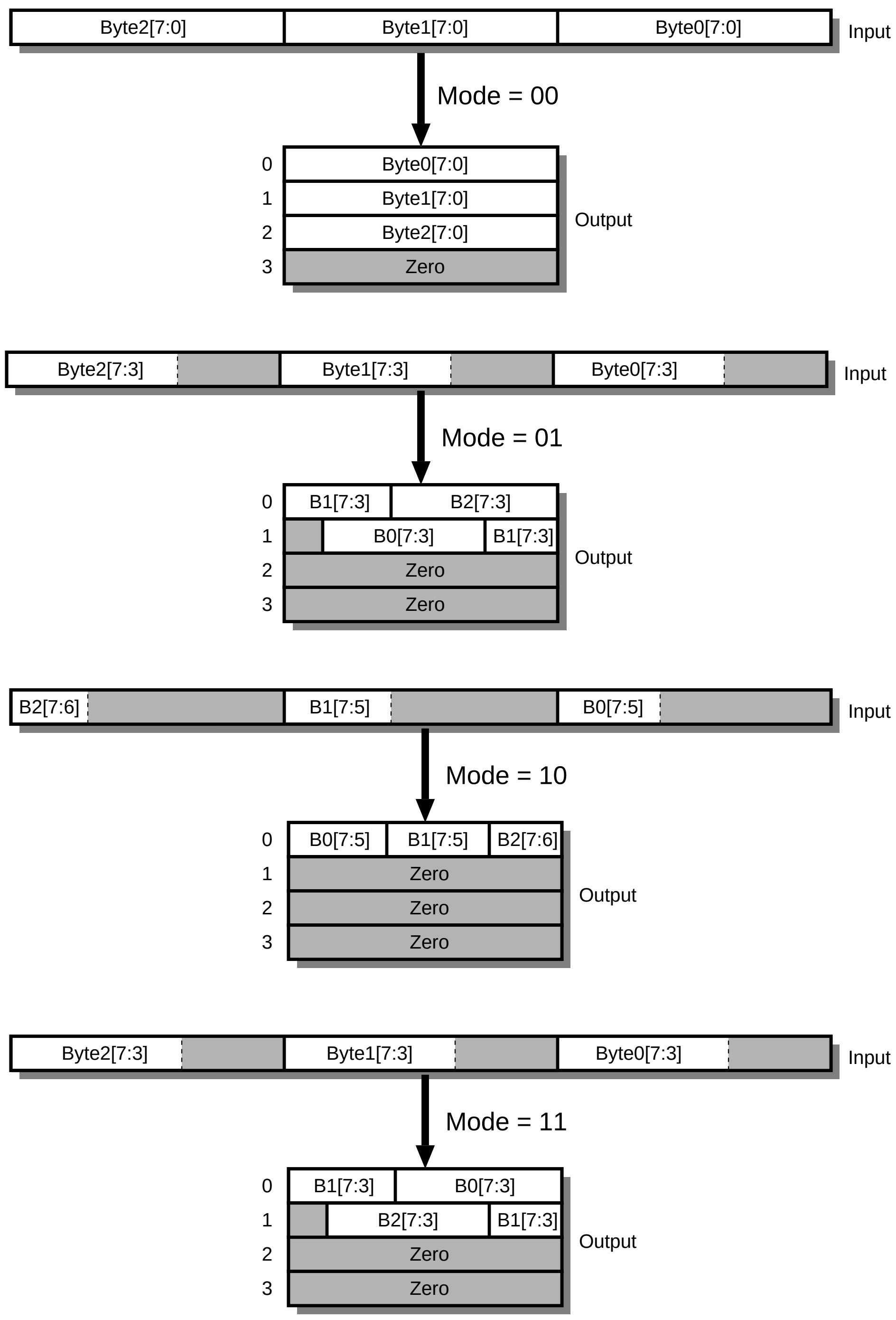

Figure 15: Effect of the MODE bits on the output pixel format. Input format is set by the Scaler

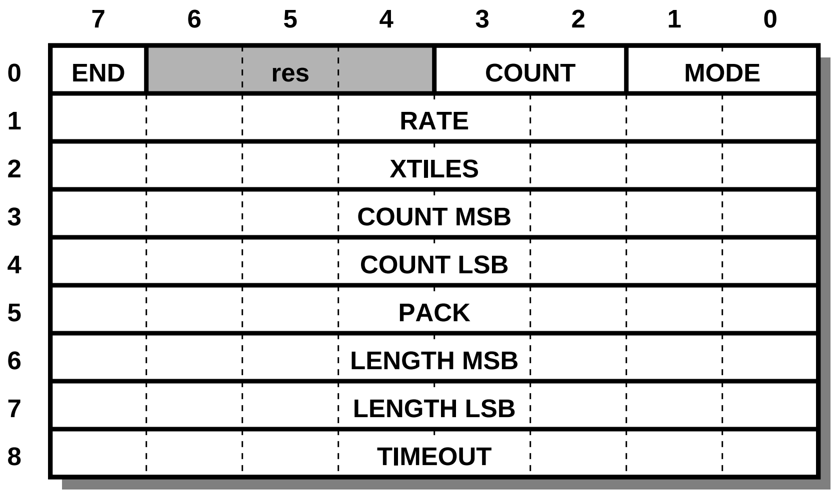

Figure 16: Control Registers.

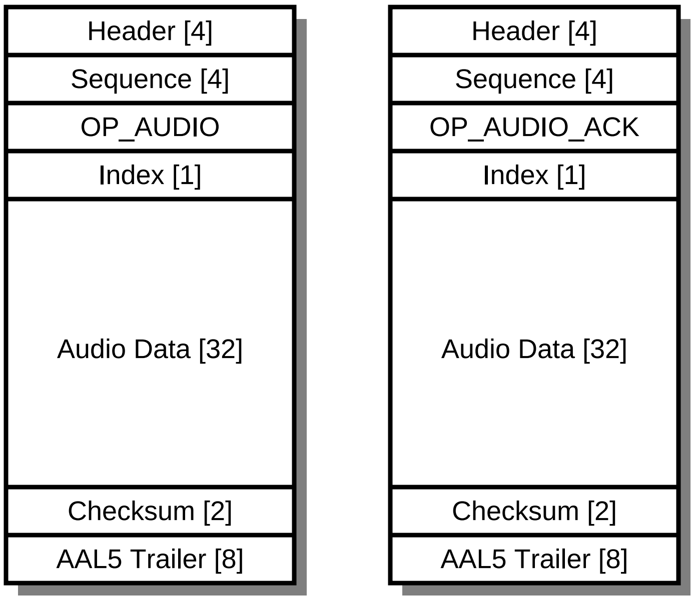

Figure 17: AVA200 Audio Load Cell.

FRB [1]
PCRB [1]
Stream Header [4]
Length [2]
Codec Regs[16]
Sequence [4]
Reserved [8]

Figure 18: AVA200 Audio data.

	7	6	5	4	3	2	1	0	
0	SOURCE		MIC	res	GAIN				Left Input
1	SOURCE		MIC	res	GAIN				Right Input
2	MUTE	res		ATTEN					AUX1 Left
3	MUTE	res		ATTEN					AUX1 Right
4	MUTE	res		ATTEN					AUX2 Left
5	MUTE	res		ATTEN					AUX2 Right
6	MUTE	res	ATTEN						Left Output
7	MUTE	res	ATTEN						Right Output
8	res	8/16	LIN/u	MO/ST	RATE			XTAL	Clock / Format
9	res		CHAN	END	res				Byte Order
A	res								Reserved
B	res								Reserved
C	res								Reserved
D	ATTEN						res	DMIX	Digital Mix
E	SAMPLES MSB.								Samples
F	SAMPLES LSB.								

Figure 19: Audio Codec Registers.

Rate	Xtal0 25.576MHz	Xtal1 16.9344MHz
0	8.0 kHz	5.5125 kHz
1	16.0 kHz	11.025 kHz
2	27.42857 kHz	18.9 kHz
3	32.0 kHz	22.05 kHz
4	Not Supported	37.8 kHz
5	Not Supported	44.1 kHz
6	48.0 kHz	33.075 kHz
7	9.6 kHz	6.615 kHz

Table 5: Audio Sample Rates

4.5.1 Codec Registers

The fields in the Codec registers have the following effects:

- Source selects the input from which to capture.
- Mic adds and extra $20dB$ of gain to the MIC input channel.
- Gain sets the gain of the input in dB.
- Mute silences a channel.
- Atten sets the attenuation of a channel in dB.
- 8/16 selects either 8 or 16 bit samples.
- Lin/μ turns on companding.
- Mo/St turns on stereo.
- Rate selects the sampling rate as shown in Table 5.
- Xtal selects the oscillator to use.
- Chan selects whether the left or right channel appears first in the PDU.
- End causes the MSB of a 16-bit audio sample to *precede* the LSB in the PDU.
- DMix turns on digital mixing of the capture data with the playback data within the chip.
- Length sets the length field of the AAL5 PDU.
- Samples specifies the number of samples per AAL5 PDU.

4.6 Schedule Load Cell Format

Sending a Schedule cell to the AVA200 causes the audio and video schedules of the unit to be updated to those specified in the cell. The AVA200 Schedule cell is structured as shown in Figure 20.

- Length specifies the number of *four* bit bucket indices in the video schedule.
- Video specifies up to 60 bucket indices for the video schedule. The most significant four bits of each octet *precede* the least significant four bits in the video schedule as shown in Figure 21. The reserved video bucket index of zero indicates an idle frame.
- Audio sets the audio buckets to be used for up to four audio streams. Currently the unit only supports a single audio stream controlled by A0. Audio bucket indices should be less than 4.

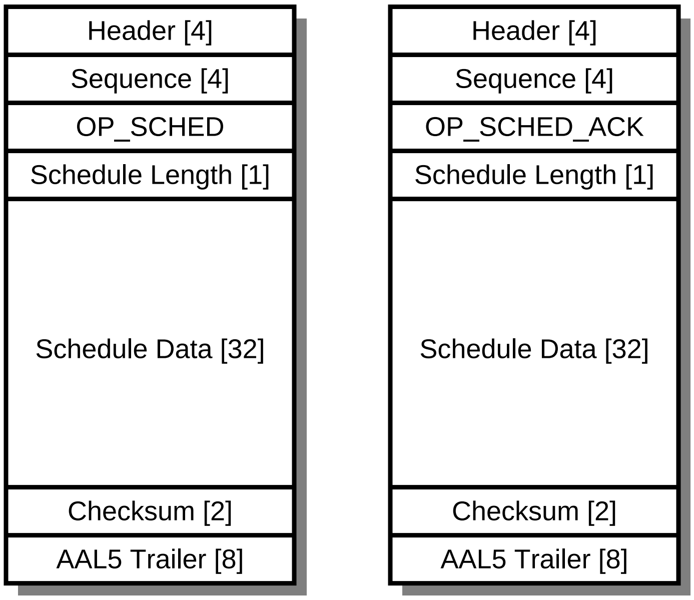

Figure 20: AVA200 Schedule Load Cell and Acknowlegement.

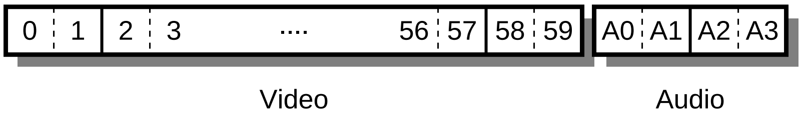

Figure 21: AVA200 Schedule Data.

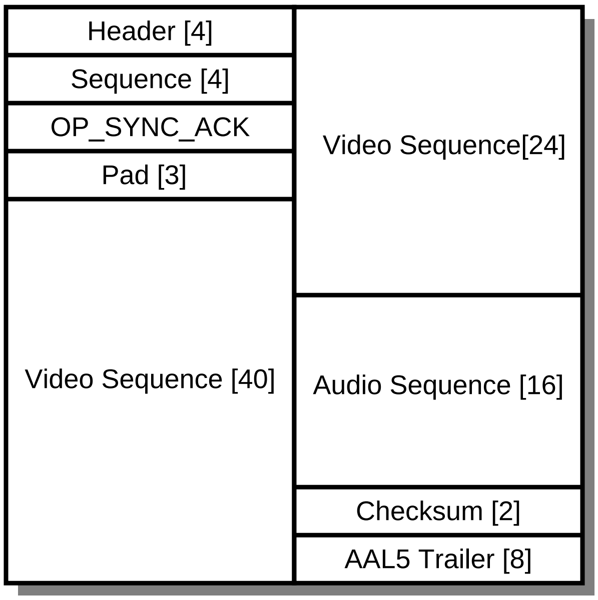

Figure 22: AVA200 Synchronisation PDU.

4.7 Synchronisation Stream Format

At the end of every shceduling period (defined by the length field of the Schedule cell), the AVA-200 sends a synchronisation PDU on the management VPI/VCI. This PDU contains the current sequence numbers of all video and audio buckets. The fields in the Sync PDU are as shown Figure 22.

- Video Sequence field contains the sequence numbers of each of the 16 video buckets. These are stored as 32-bit *big-endian* unsigned integers.
- Audio Sequence field contains the sequence numbers of each of the 4 audio buckets. These are stored as 32-bit *big-endian* unsigned integers.

This feature is not present on some early versions of the unit.

5 Network Video and Audio Formats

Both the audio and video streams are sent encapsulated within AAL5 frames. The video and audio service specific convergence sub-layers are described below.

5.1 Uncompressed Video Format

The AVA-200 does not transmit the digitised video in scanline format. Instead, a frame is broken down into tiles each of which is 8x8 pixels in size. Each tile is encoded on the ATM network in scan line order (each scan line comprising eight pixels). For an 8 bit RGB stream each scan line will comprise 8 octets. The mapping is illustrated in Figure 23.

The AVA-200 can pack a variable number of tiles (definable on a per stream basis) into a single AAL5 frame. Note that when asking the AVA-200 to pack multiple tiles into a single frame the packing factor is required to be set such that the picture frame requested encodes to an integral number of fully packed AAL5 frames. Tiles are packed contiguously the final one being followed by a variable length pad, a 8 octet tile trailer and finally an 8 octet AAL5 trailer. The pad is used such that the AAL5 PDU frame is an integral number of cells (48 bytes) in length. The tile trailer

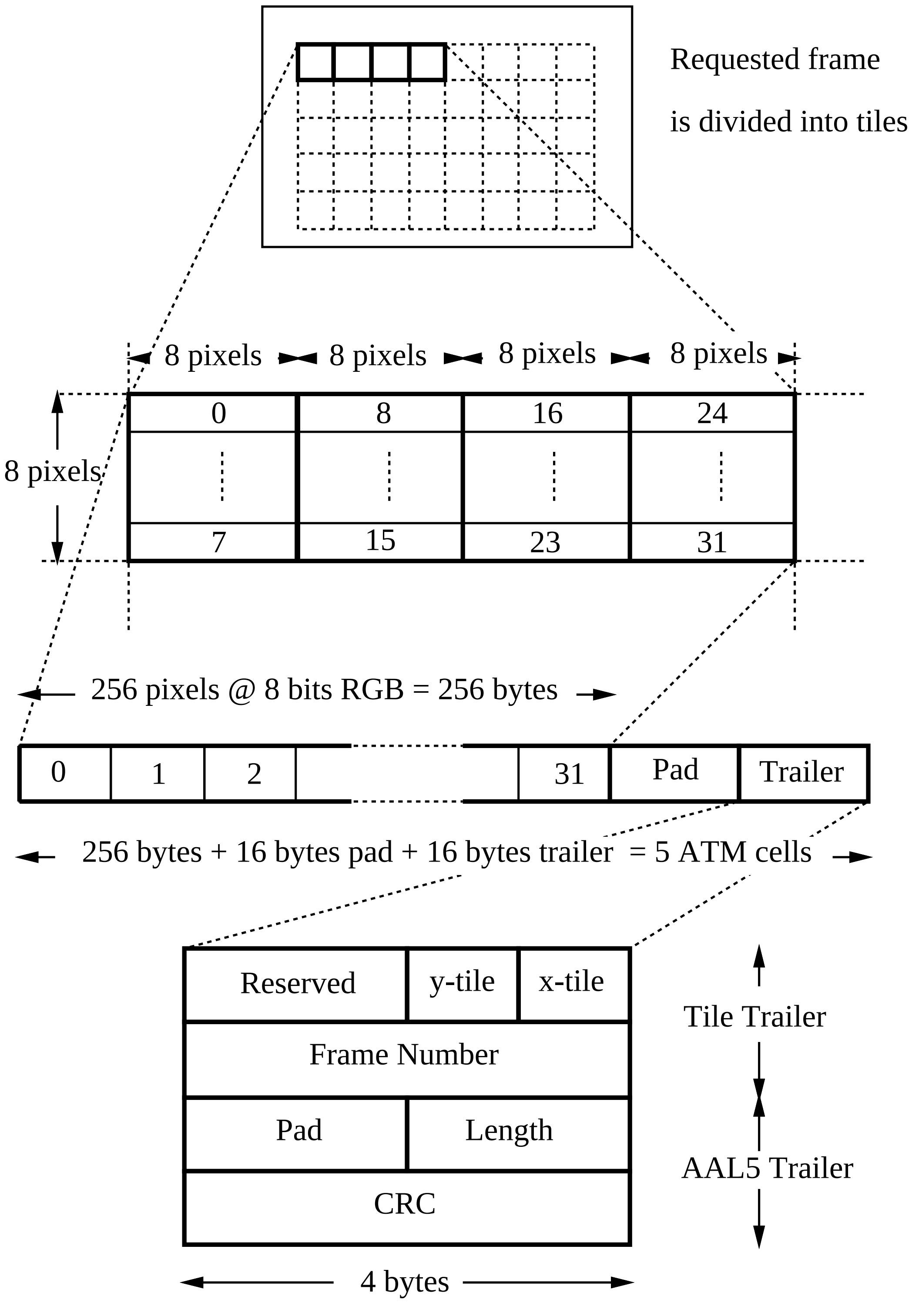

Figure 23: 8 Bit RGB Video Tile Encoding

contains the co-ordinates of the first tile in tile dimension units. Also present is a 32 bit picture frame number to which this tile belongs.

Figure 23 illustrates the AAL5 frame format when an 8 bit RGB stream is encoded with a packing factor of four. As illustrated the packing order is left to right, when one row of tiles has been encoded the next row is started. It is acceptable for the packing factor to be set such that tiles from multiple rows be encoded in the same AAL5 PDU. Indeed, if space permits a whole frame may be packed into a signle AAL5 PDU.

5.2 JPEG Compressed Video Format

The AVA-200 supports JPEG video compression as an option. If a JPEG stream is selected from a suitably equipped AVA then the encoding on the ATM network is slightly different from the uncompressed coding previously described. The picture frame is still split into and transmitted as a sequence of tiles except that the compressed pixel data will be of varying size. The size of the compressed result will depend on the composition of the original tile (typically unknown) and the JPEG compression parameters selected.

The AVA-200 JPEG video encoding is illustrated in Figure 24 (again a packing factor of four is used as an example). Note that if multiple JPEG compressed tiles are to be packed in a single AAL5 frame then it is necessary to unpack the tiles by using the **RST Offset** field in the final cell to backstep through the AAL5 frame. Note also that the packing factor for JPEG compressed video should be set carefully. If the packing factor is set too high then the AAL5 frame produced by the AVA-200 for a pessimal case sample may overflow the maximum allowable in the sink entity.

5.3 Audio Format

In a manner similar to the video encoding the AVA-200 is able to pack a variable number of samples into a single AAL5 SDU. The ATM encoding for 16-bit stereo audio from the AVA-200 is illustrated in Figure 25. For stero encoding the left channel data is always transferred before the right channel data. The number of bytes that the AVA-200 will buffer before generating a trailer is controllable by the remote client. Note that the total audio sample data length selected must be a multiple of 16 octets. The figure chosen is a tradeoff between a variety of concerns. If a small number is chosen, e.g. one sample per frame, then a great number of cells will be generated, each with a high header/padding overhead. For example, 16-bit stereo sampled at 44.1 kHz will require 18.3 Mbps of network bandwidth even though the sample bandwidth is only 1.4 Mbps. A stock workstation would find great difficulty sinking such a stream given the high rate of AAL5 frames that would have to be transferred through the operating system. A dedicated hardware sink would have less difficulty. This is analogous to the tradeoff in the AVA-200 video encoding case. Note that when multiple samples are buffered then an audio cell is sent as soon as it is full.

6 AVA200 Console Interface

6.1 Power On Reset

After a power on reset, console output should appear as showin in Figure 26 console interface:

When the ATM transmission interface is in violation (indicated by the Blue LED permanently on), it is normal for the console output to stop when the "`Protocol Init`" stage is reached. As soon as the violation is cleared console output should continue.

6.2 Diagnostics

During normal operation of the unit, console output should contain messages of the following form:

- `Proto: Bad cell`
  `Cell Rx: 01DEADBEEF0102030405060708090A0B0C0D0E0F10111213141516`

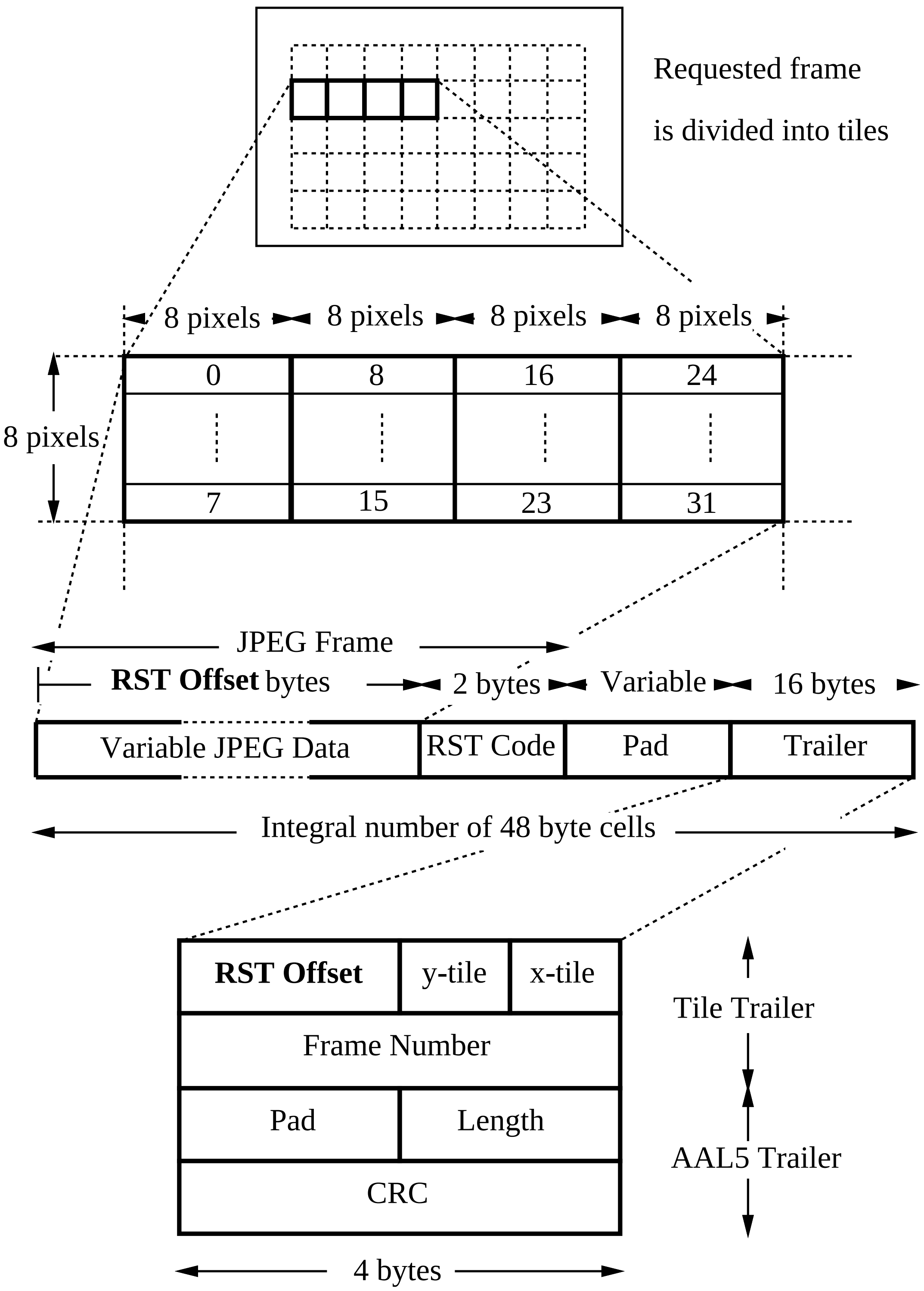

Figure 24: JPEG Video Tile Encoding

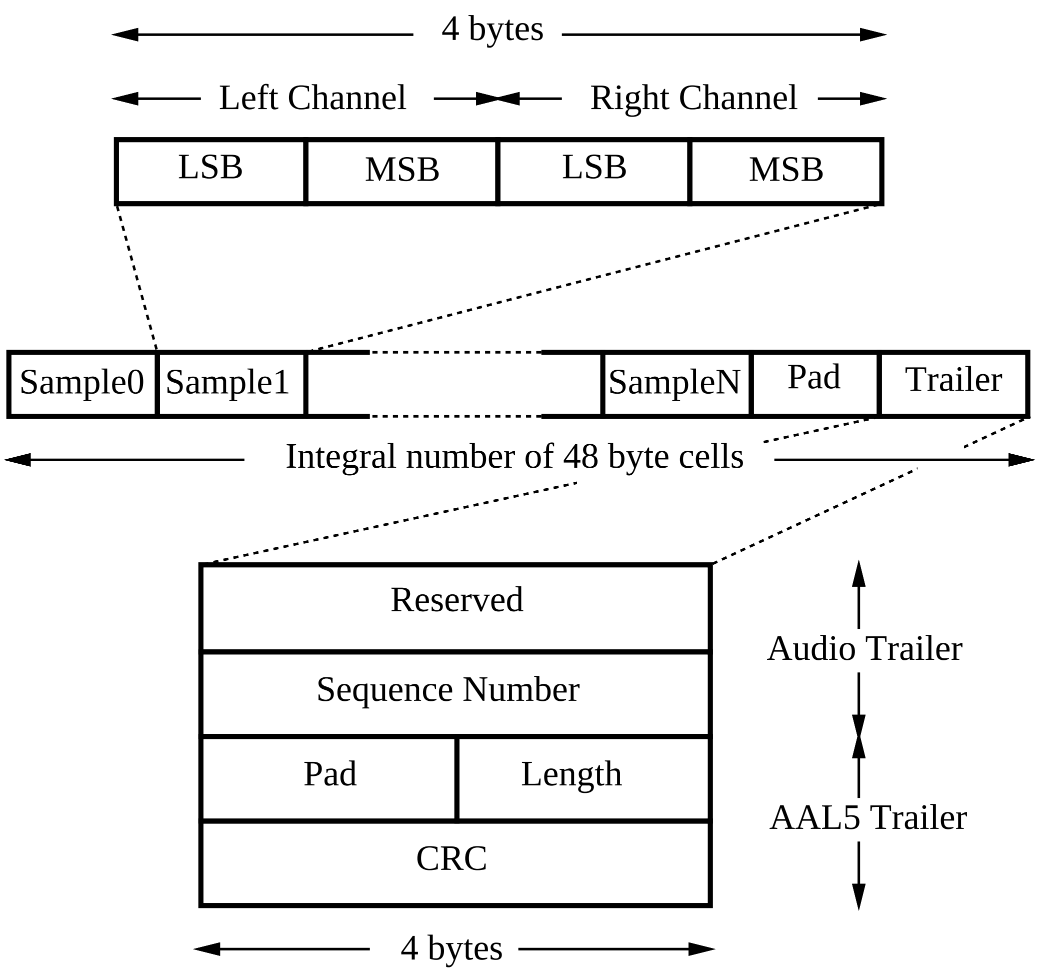

Figure 25: 16-bit Stereo Audio Encoding

```
(c) 1993 Nemesys Research Ltd.
Device Version 200.0
Firmware Version 1.1
I2C Driver Init... OK
Xilinx Init... OK
ATM Init... OK
SAA7191 Init... OK
SAA7186 Init... OK
C-Cube Init... OK
AD1848 Init... OK
Timer Init... OK
Protocol Init... OK
Scheduler Init... OK
```

Figure 26: AVA200 Power On Diagnostics.

`1718191A1B1C1D1E1F2021222324252627000000028BF671ED0`
Indicates that an unrecognised cell was received before the management header is known, or that the cell failed the 16-bit checksum.

- `Proto: Reset FFFF00000080`
  Output when the unit receives a valid reset cell. The HEX number is actually the FRB, PCRB and HEADER of the management stream. The reset diagnostics will be repeated as above.

- `Proto: Duplicate cell`
  Indicates that a protocol cell was received whose sequence number was identical to the previous cell.

- `Proto: Video 0A`
  Indicates that video bucket 0x0A has been updated.

- `Proto: Audio 01`
  Indicates that audio bucket 0x01 has been updated.

- `Proto: Schedule Length 19`
  Indicates that a schedule of length 0x19 has been loaded.

- `Proto: Sync`
  Output whenever a synchronisation PDU is generated.

6.3 Main menu

Pressing any key will cause the scheduler to be exited and the AVA200 Debug Monitor to be entered. The prompt “>” will be displayed. The AVA200 Debug Monitor will display a list of the currently acceptable commands whenever the “?” character is typed. The main menu should appear similar to Figure 27.

The commands have the following effects:

`S` Displays the contents of the status registers for the main functional units of the AVA200.

`8` Moves control to the SAA7186 Scaler sub-menu.

`9` Moves control to the SAA7191 Decoder sub-menu.

```
Commands:
S  Status
A  ATM Interface Tests
8  SAA7186 Scaler Tests
9  SAA7191 Decoder Tests
C  AD1848 Audio Codec Tests
J  C-Cube (JPEG) Tests
X  Xilinx Tests
I  Probe I2C bus
R  Remote Mode
P  Protocol on/off
M  Memory Dump
T  TAXI Reset
?  Print help message
```

Figure 27: AVA200 Main Menu.

`C` Moves control to the AD1848 Codec sub-menu.

`J` Moves control to the C-Cube JPEG sub-menu.

`X` Moves control to the Xilinx sub-menu.

`I` Scans all addresses on the I^2C bus and reports any devices found.

`R` Enters "Remote Mode" by restarting the protocol stack and the scheduler.

`P` Toggles whether cells received over the ATM Transmission interface will be passed to the protocol stack.

`M` Prompts for a 16-bit HEX address and dumps 32 bytes of SRAM or Xilinx registers starting from that address.

`T` Resets the TAXI chips of the ATM Transmission Interface.

`?` Displays the help menu shown above.

Any other key will cause the response: "`Unrecognized command.`"

6.4 ATM Interface Tests

When ATM menu is entered, the prompt changes to "`ATM>`". The commands available from the ATM menu are shown in Figure 28.

The commands have the following effects:

`I` Initializes the ATM interface and resets the Rx FIFOs.

`T` Transmits test cells over the ATM interface. Test cells have header "`DEADBEEF`", ascending payload, AAL5 length of "`0028`" and AAL5 CRC32 of "`BF671ED0`".

`R` Receives and displays cells over the ATM interface.

`B` Transmits test cells once a second and displays all received cells. This is intended for use using a fibre loopback.

```
Commands:
I  Initialize
T  Transmit Cells
R  Receive Cells
B  Bidirectional (Tx/Rx)
L  Loopback Test
F  Fabric/Taxi toggle
E  Edit route bytes
?  Print help message
```

Figure 28: AVA200 ATM Menu.

`L` As above, but transmits faster and automatically checks the received cells against the expected test cells reporting any errors.

`F` Selects whether to use the transmission interface or the fabric connection (where fitted).

`E` Edits the route bytes transmitted when the fabric interface is selected.

`?` Displays the help menu shown above.

Pressing "`ESC`" will return to the top-level menu. Any other key will cause the response: "`Unrecognized command.`"

6.5 SAA7186 Scaler Tests

When the Scaler menu is entered, the prompt changes to "`SAA7186>`". The commands available from the scaler menu are shown in Figure 29.

```
Commands:
I  Initialize
S  Status
R  Display register values in RAM
E  Edit register values in RAM
W  Write register values to SAA7186
?  Print help message
```

Figure 29: AVA200 SAA7186 Scaler Menu.

Pressing "`ESC`" will return to the top-level menu. Any other key will cause the response: "`Unrecognized command.`"

6.6 SAA7191 Decoder Tests

When the Decoder menu is entered, the prompt changes to "`SAA7191>`". The commands available from the decoder menu are shown in Figure 30.

Pressing "`ESC`" will return to the top-level menu. Any other key will cause the response: "`Unrecognized command.`"

```
Commands:
I  Initialize
S  Status
R  Display register values in RAM
E  Edit register values in RAM
W  Write register values to SAA7191
?  Print help message
```

Figure 30: AVA200 SAA7191 Decoder Menu.

6.7 AD1848 Codec Tests

When the Codec menu is entered, the prompt changes to "`AD1848>`". The commands available from the codec menu are shown in Figure 31.

```
Commands:
I  Initialize
S  Status
R  Display register values
W  Write register value
M  Mode change enable
F  Mode change finished
L  Loopback test
P  Playback test
8  8KHz Mono 8-bit u-law
4  48KHz Stereo 16-Bit linear
?  Help
```

Figure 31: AVA200 Codec Menu.

The commands have the following effects:

`I` Initializes the AD1848 Codec chip.

`S` Read the status register of the codec.

`R` Displays the current register values.

`W` Write to one of the registers (see data book.)

`M` Initiate a mode change. Certain registers may only be updated during a "mode change".

`F` End a "mode change".

`L` Capture and playback samples using a 4K circular buffer.

`P` Playback a 8KHz sample stored in the EPROM. This sample is omitted from most boards.

`8` Set sample format and rate.

`4` Set sample format and rate.

`?` Display the help message.

Pressing "`ESC`" will return to the top-level menu. Any other key will cause the response: "`Unrecognized command.`"

6.8 C-Cube CL550 Tests

When the C-Cube menu is entered, the prompt changes to "`C-Cube>`". The commands available from the C-Cube menu are shown in Figure 32.

```
Commands:
I  Initialize
S  Status
R  Read register value
W  Write register value
M  Multiple reads
D  Dump registers
G  Go
C  Compression Test
F  Fast Drain Test
?  Help
```

Figure 32: AVA200 CCube Menu.

The commands have the following effects:

`I` Initializes the CCube chip. This sets up all necessary registers but does not reload the Huffman Tables or Quantizer tables. These are only loaded after a hard-reset.

`S` Read the status register of the CCube.

`R` Displays a register value.

`W` Write to one of the registers (see data book.)

`M` Read a register many times in a tight loop until a key is pressed.

`D` Dump the next 32 registers startng at the last one read.

`G` Start a CL500 compression cycle.

`C` Read and display the compressed JPEG data from the CL500 Codec register.

`F` Drain JPEG compressed data from the CL550 as fast as possible.

`?` Display the help message.

Pressing "`ESC`" will return to the top-level menu. Any other key will cause the response: "`Unrecognized command.`"

6.9 Xilinx Tests

When the Xilinx menu is entered, the prompt changes to "`Xilinx>`". The commands available from the Xilinx menu are shown in Figure 33.

The commands have the following effects:

`I` Initializes the Xilinx chip. This also resets the auxiliary chips.

`S` Read the status register of the Xilinx.

```
Commands:
I  Init
S  Status
C  Current register values
L  Last register values
W  Write registers
T  Test LEDs
1  IAP Test 1
2  IAP Test 2
3  IAP Test 3
?  Help
```

Figure 33: AVA200 Xilinx Menu.

`C` Displays current register values of readable registers.

`L` Displays value last written to a register.

`W` Write to one of the registers (see H/W documentation!)

`T` Do a lamp test to ensure LEDs are OK - only possible if the Xilinx has programmed correctly.

`1` Hardware debugging test 1 - **DO NOT USE**

`2` Hardware debugging test 2 - **DO NOT USE**

`3` Hardware debugging test 3 - **DO NOT USE**

`?` Display the help message.

Pressing "`ESC`" will return to the top-level menu. Any other key will cause the response: "`Unrecognized command.`"