



PlayCable- A Technical Summary

Version: 2019-07-25

Contents

1	Acknowledgements.....	3
2	Introduction	3
3	PlayCable System Architecture	4
4	Head-End Hardware and Software	6
4.1	DataChannel Transmitter Card	6
5	PlayCable Adapter Architecture.....	8
6	PlayCable Adapter Hardware.....	10
6.1	Digital Subsystem.....	12
6.2	PlayCable ASIC.....	14
6.2.1	Master Component Interface	14
6.2.2	Bootstrap ROM	15
6.2.3	PlayCable RAM Interface.....	15
6.2.4	PLL Tuner Interface Module.....	15
6.2.5	Serial Receiver Interface Module.....	15
6.3	RAM.....	16
6.4	RF Cable Receiver.....	17
6.5	PLL Tuner.....	17
6.6	Demodulator.....	19
6.7	Power Supply	20
7	Software.....	21
7.1	PlayCable Bootstrap ROM.....	21
7.2	PlayCable Menu Program	25
7.3	PlayCable Game ROMs.....	28
8	PlayCable Adapter Variations	30
8.1	Jerrold PlayCable.....	30
8.2	Jerrold Engineering Audio PlayCable	37
8.3	Joe Jacobs / Dennis Clark Development System PlayCable	38
8.4	Intellivision 2 Compatibility Modification	42
9	PlayCable Phoenix.....	45

1 Acknowledgements

The work summarised in this document would not have been possible without the valuable contribution of Tom Boellstorff, carlsson, Dennis Clark, Charles Dages, dave1dmarx, Paul Hilt, hoalst, Joe Jacobs, intvnut, intvsteve, Knarfian, Lathe26, Ron The Cat, Braxton Soderman, Splurge and Gary Stein. My thanks to all of them for their help.

decle 2019-07

2 Introduction

PlayCable was a service and peripheral for the Mattel Intellivision developed by General Instrument's cable subsidiary, Jerrold, in the late 1970s. In a joint venture between General Instrument and Mattel, it became the first commercial subscription service to offer downloadable video games when it was officially launched in 1981. The service ran in a number of US cities until its closure in 1983.

This document describes the technical implementation of the General Instrument PlayCable service. It primarily focuses on the PlayCable adapter peripheral, but also briefly describes the head end infrastructure used to broadcast games. The description of the PlayCable adapter has been constructed from an examination and partial reverse engineering of a 236-817 REV B PlayCable PCB. Photographs of the internals of an example of a Jerrold branded version of the PlayCable adapter have also been examined. Based on the styling of the PlayCable logo, trademark filings and implementation details, the Jerrold design is believed to be the earlier of the two models. The differences between the Jerrold and General Instrument machines are detailed in Section 8.1.

3 PlayCable System Architecture

The PlayCable adapter forms one part of the larger PlayCable service, comprising a head end system located at the Cable TV provider, the cable infrastructure and the Intellivision PlayCable adapter located in a customer's home.

An overview of the full system is provided in two contemporary documents available in the public domain. Both were written by Charles L Dages, the PlayCable Product Manager at Jerrold.

- [Consumer Software Services Via Cable Television Systems](#)
- [PlayCable: a Technological Alternative for Information Services](#)

These documents show that Jerrold saw the transmission of games as only one application for DataChannel, the cable technology that underpinned the PlayCable product. This view is reinforced by a marketing paper also presented at the 1980 NCTA convention.

- [Jerrold DataChannel the Optimal Information Delivery Technology for Cable](#)

The Dages documents also outline how a PDP-11 minicomputer, stocked with custom data channel cards, was used to generate a number of broadcast channels, conceptually similar to FM radio stations. Each channel contained one or more programs and was transmitted through the cable infrastructure, as shown in Figure 1.

The PDP-11 minicomputer managed the suite of data channel cards, configuring them with the programs to be transmitted and monitoring their performance. The data channel cards operated independently, using their own RAM and CPU to store and transmit the data for two channels. Each program channel could support multiple games by transmitting an identifying header in front of every program. Clients would then selectively download a specific game having matched the program name in the header. Information about which stream contained a particular program was made available through a menu program which was transmitted on a pre-determined "Catalog" stream.

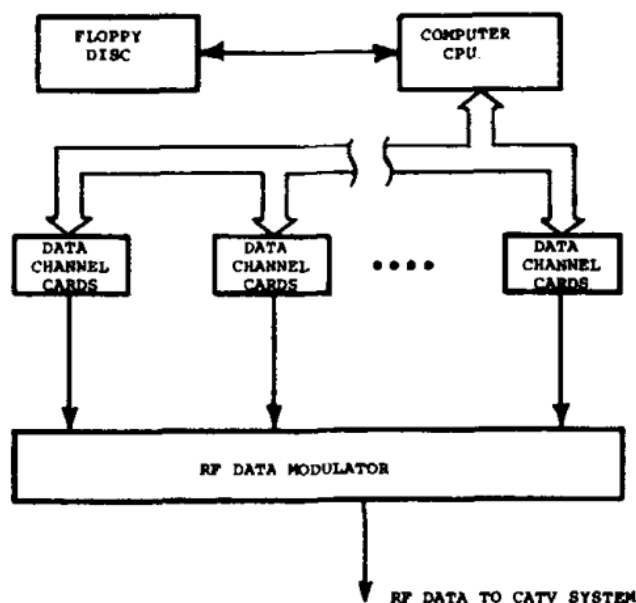


Figure 1 : PlayCable Head End Architecture

The distributed and independent nature of the head end, as well as the fact that each channel card delivered two channels, contributed to the user experience of the monthly changeover of the game catalogue, as reported in "[Memories of PlayCable](#)" by Michael Roode:

"At around 8:30 a.m. the games would vanish from the menu. It would still boot up, and there would still be a "PlayCable Presents" menu title screen, but when you pressed the disc nothing happened. Then every couple minutes or so, the games would pop back on the menu two at a time until the entire menu was refilled, albeit with a couple different games."

This would suggest that at the switch-over the entries for all channel cards were removed from the menu. Then, as the new ROM images were incrementally loaded to the channel cards by the PDP-11, the games would be made available via the menu program.

It is known from conversations with ex-PlayCable Company employees that, whilst the scheduling of which games would be broadcast each month was undertaken by the PlayCable Company, the preparation and distribution of games to cable companies was completed by Jerrold. As a consequence, there was no differentiation between affiliates in terms of branding and all cable companies broadcast the same library of games each month.

4 Head-End Hardware and Software

Very little information, other than the high-level architecture documents detailed in Section 3, has been found regarding the hardware and software used to generate PlayCable data streams within the cable company head-end infrastructure. The configuration and integration of the PDP-11 host is unknown.

4.1 DataChannel Transmitter Card

The one piece of specific head-end infrastructure that has turned up is an example of a DataChannel transmission card which was used to format the digital streams ready for transmission. A number of these were hosted in a PDP-11 minicomputer and each was responsible for the generation of two broadcast streams of program data. An example of one of these DCX11A transmission cards is shown in Figure 2. The DCX11A is a dual DEC QBUS card, manufactured by Jerrold and suitable for a later model PDP-11.



Figure 2 : Jerrold DCX11A DataChannel Transmission Card

Looking at the image it can be seen that the DCX11A DataChannel card is a small single board computer based on an MC6808 microprocessor, the large 40-pin chip at the upper centre of the image. The MC6808 is a member of the Motorola 6800 8-bit microprocessor family with internal clock support circuitry. The frequency of the crystal at the top of the board is not known, however, the MC6808 datasheet suggests that it is probably between 1MHz and 4MHz, resulting in the processor running at between 250KHz and 1MHz. The CPU is backed by a windowed 2716 2K x 8-bit EPROM immediately below the processor, and 32K of RAM held in two banks of eight 16K x 1-bit DRAM chips to the left of the board. RAM is implemented using a mixture of MK4215 and MC4116 chips. It is interesting that the window of the EPROM has been left uncovered, and as a consequence, there is a reasonable chance that the board's firmware will have been lost as a result of UV exposure since its manufacture, which is sometime after August 1980 judging by chip date codes.

It is believed that communication with the host PDP-11 is managed by the MC6821P Parallel Interface Adapter (PIA), the 40-pin chip at the bottom right of the image. Finally, two MC6850P Asynchronous Communication Interface (ACI) chips, seen below the EPROM at the bottom of the

image, are understood to convert the parallel program data stored in RAM into serial broadcast streams. Each of the two ACIs is responsible for generating a single broadcast stream and feeds its output to one of the two gold coaxial connectors at the bottom left of the card, as shown in Figure 3.

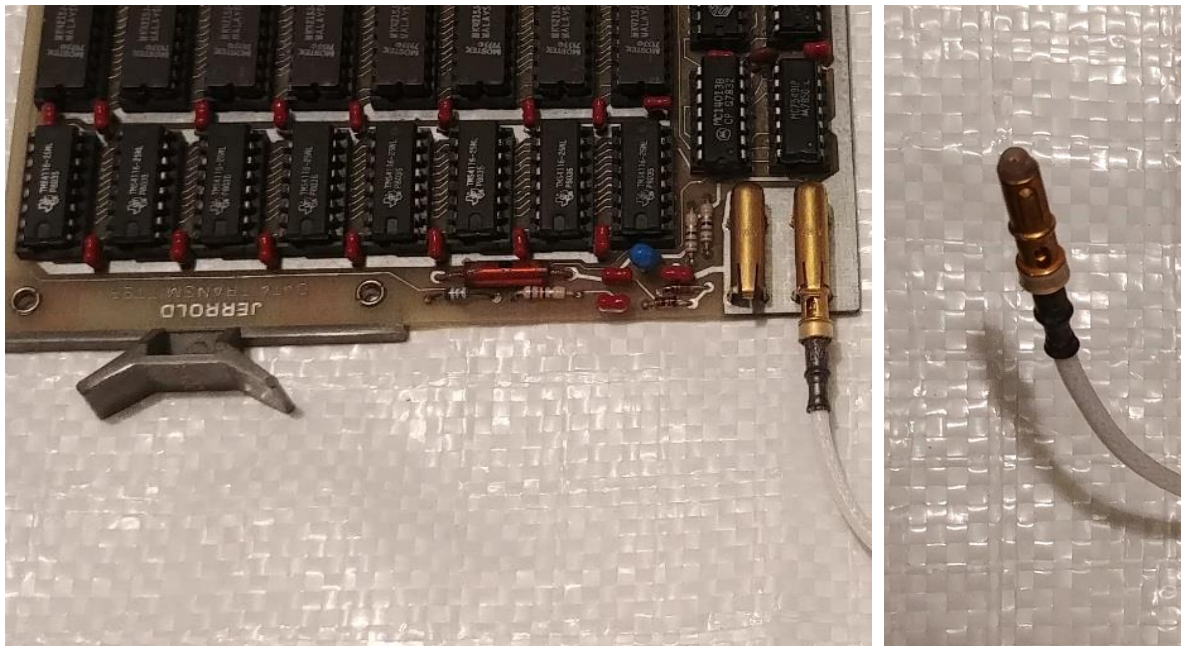


Figure 3 : Jerrold DCX11A DataChannel Coaxial Output

The remaining ICs on the board are standard glue logic chips, believed to be providing address decoding, QBUS interface logic and DRAM refresh signals. It is suspected that the 4 DIP switches seen in the centre of the board are used to provide the card's identity on the QBUS, probably by mapping it to different address ranges in PDP-11 memory. If correct, this would allow up to 16 DataChannel cards to be hosted in a single PDP-11, making each host capable of generating up to 32 broadcast channels.

5 PlayCable Adapter Architecture

As already noted, the component of the PlayCable service deployed in customers' homes was the Intellivision adapter peripheral. This was an add-on owned and supplied by the cable company that plugged into the cartridge port of the customer's Intellivision Master Component. It was approximately half the size of the Master Component and styled to match the Intellivision I. In addition to the Intellivision connection, the PlayCable adapter required a feed from the cable TV signal, which it obtained from a dedicated switch box, and a mains power connection.

The architecture of the PlayCable adapter is made up of six major components as summarised in Figure 4.

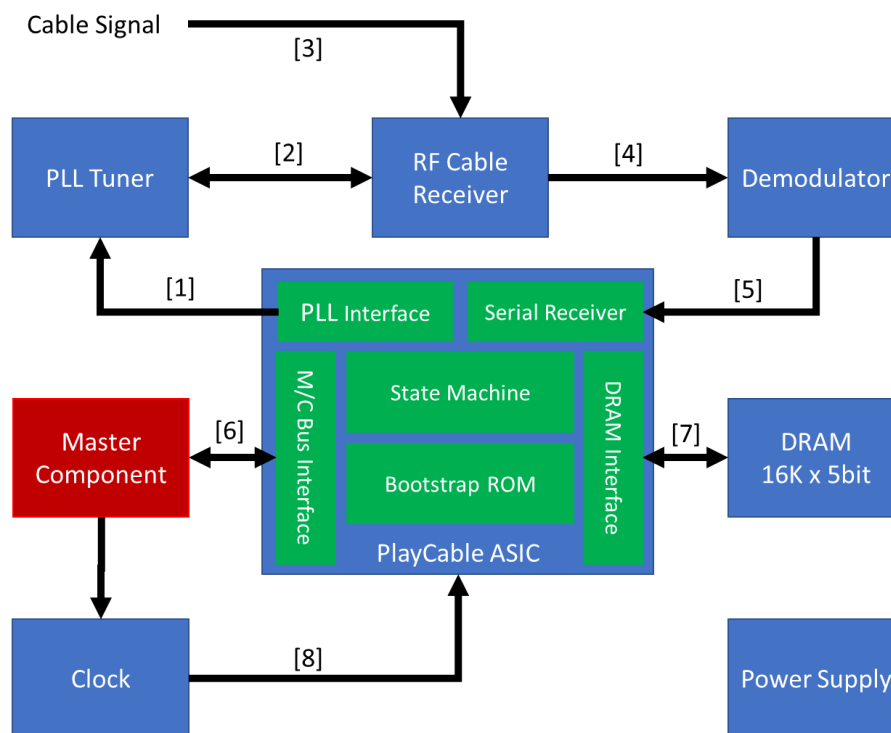


Figure 4 : PlayCable Adapter Architecture

The Application Specific Integrated Circuit (ASIC) at the heart of the machine configures a digital phase lock loop (PLL) tuning circuit [1]. The PLL sets the frequency of a radio frequency (RF) cable receiver [2]. The cable receiver takes the incoming cable feed [3], filters the appropriate channel out of it and removes the RF component. The resulting frequency modulated (FM) signal is fed to a demodulator [4] that extracts the embedded digital signal which is, in turn, passed back to the ASIC for interpretation [5].

The PlayCable ASIC makes the digital information embedded in the stream available to the Intellivision Master Component at known addresses in its memory map [6]. The ASIC will also accept and fulfil requests from the Master Component to read or write data to the PlayCable's internal DRAM storage [7]. All accesses to the PlayCable memory are the result of requests from the Master Component. The PlayCable does not transfer data directly from the incoming data stream to its RAM. A small clock circuit takes the MCLK signal from the Master Component cartridge port and cleans it up before passing it on to the ASIC [8].

The PlayCable ASIC also contains a small bootstrap program in ROM. When it is executed this sets the PLL tuner to a known frequency and attempts to load a predefined program. The program downloaded typically contains a menu to select and load games, although the adapter does not restrict it to this purpose. While the bootstrap ROM does contain subroutines that might be useful to the menu program, it does not contain the menu program itself.

The final component of the PlayCable adapter is a power supply that takes a mains AC supply and generates the required DC voltages from it.

6 PlayCable Adapter Hardware

The following section describes the various hardware components of the PlayCable. Images of the PlayCable PCB are shown in Figure 5 and Figure 6.

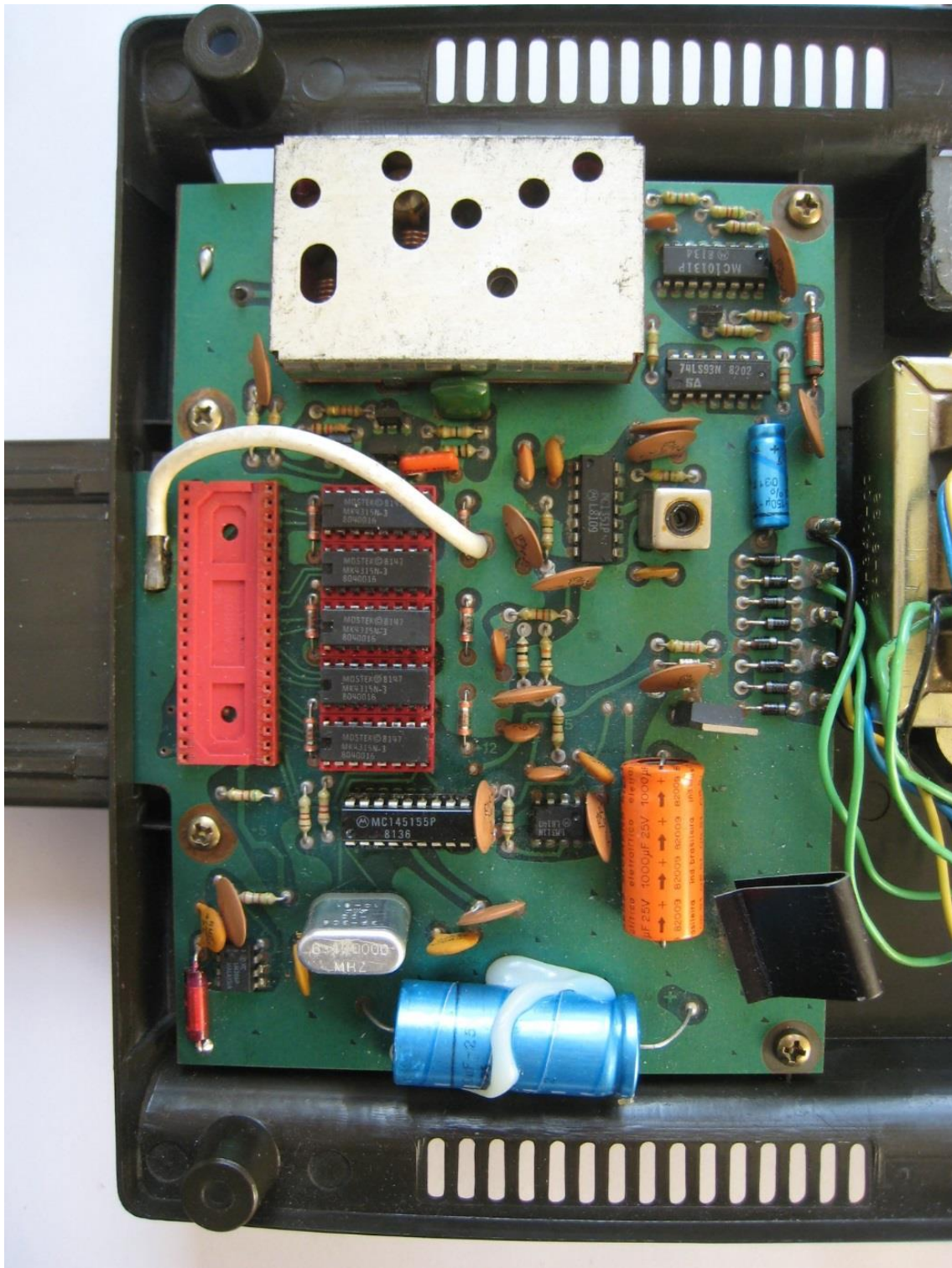


Figure 5 : PlayCable PCB Component Side

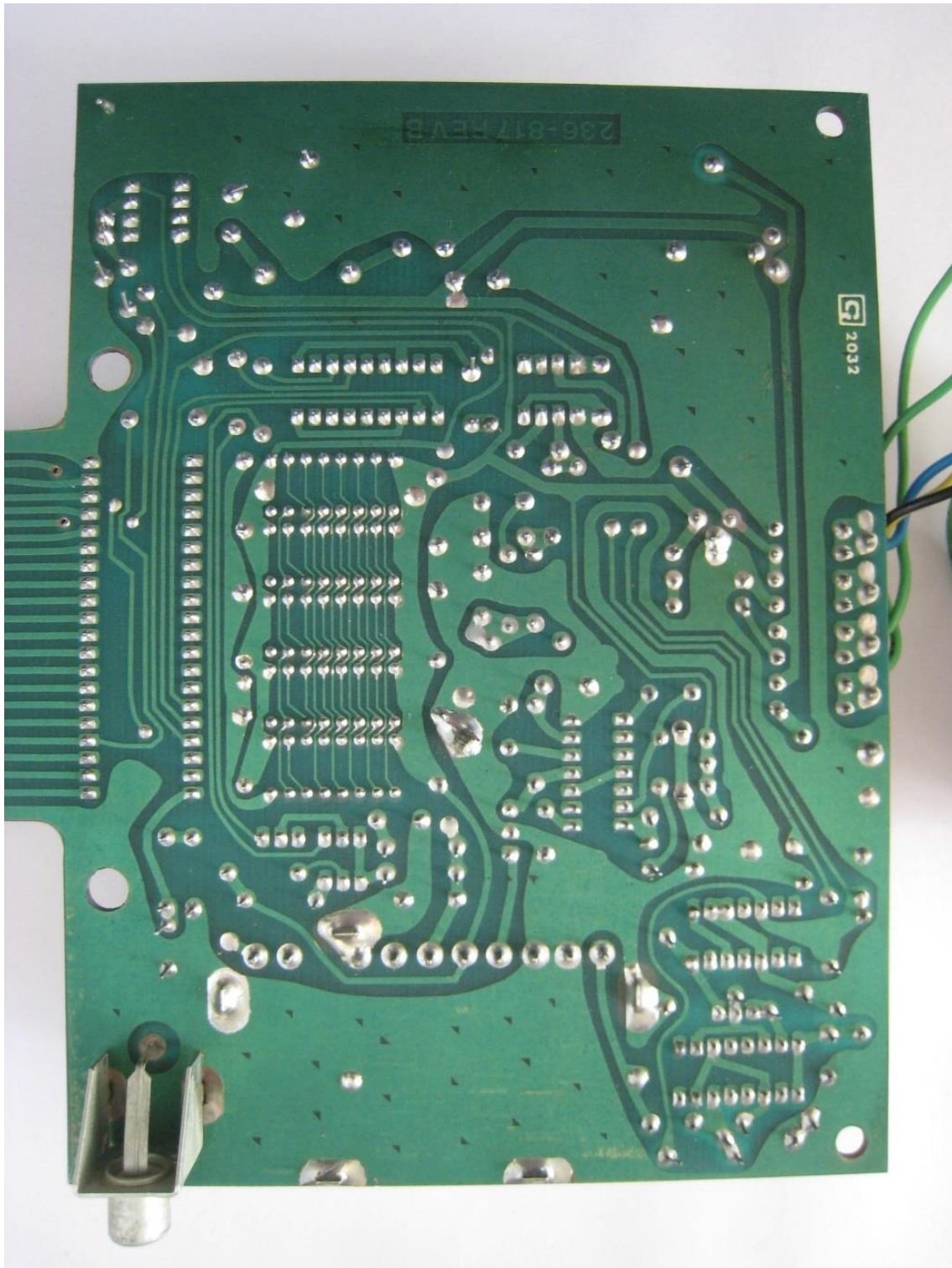


Figure 6 : PlayCable PCB Solder Side

Only four case screws need to be removed to expose the PCB. Unlike other Intellivision products, the PlayCable PCB is not encased in an RF shield.

The mapping of the architecture shown in Figure 4 to the physical board layout is shown in Figure 7.

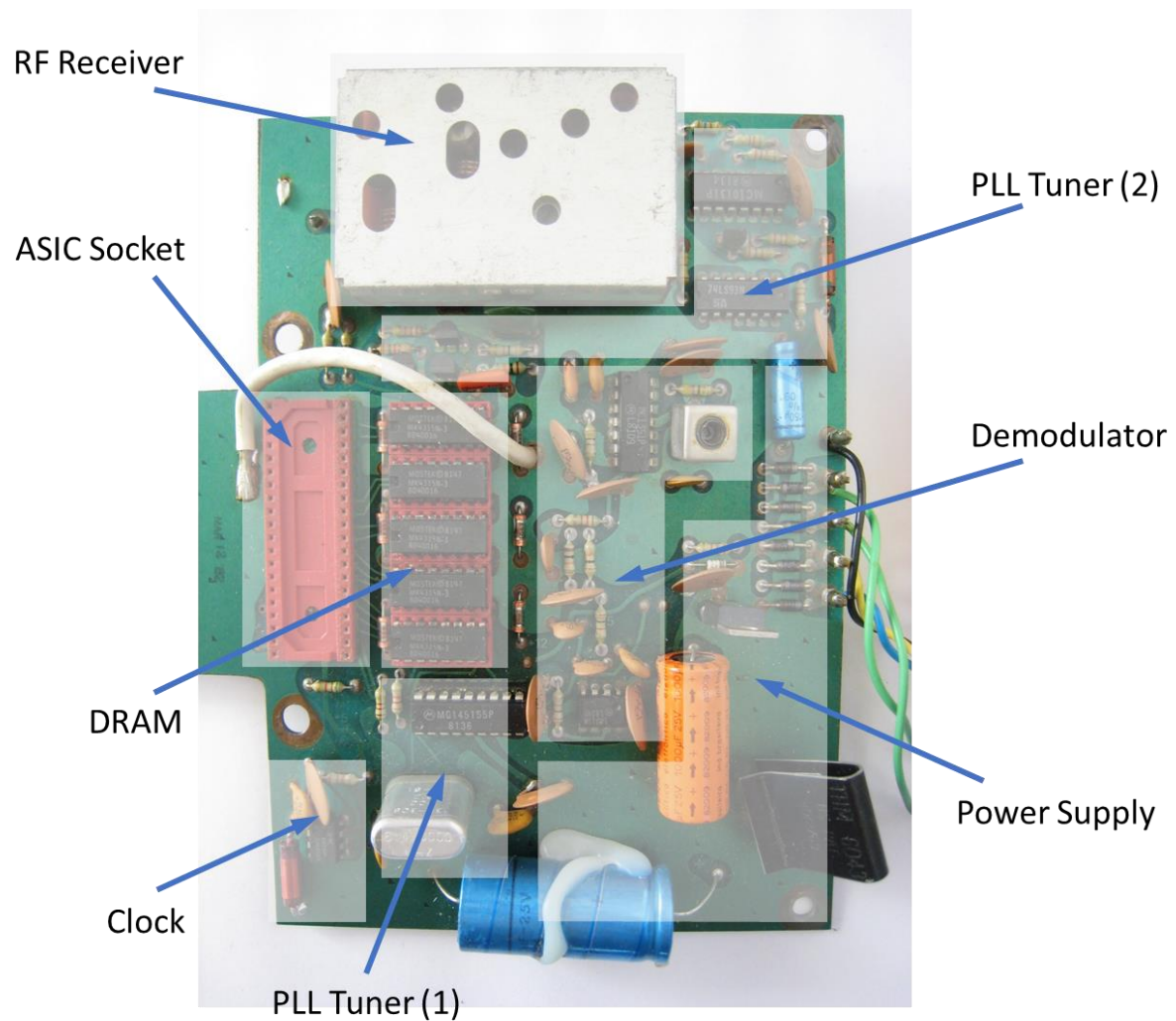


Figure 7 : PlayCable PCB Architecture

Note that although the PLL tuner is split into two locations on the board (PLL Tuner 1 and PLL Tuner 2) it operates as a single logical entity.

6.1 Digital Subsystem

At the core of the PlayCable is a relatively self-contained digital subsystem comprising the Intellivision Master Component cartridge interface, the PlayCable ASIC, the game storage RAM and a clock circuit. Figure 8 shows the schematic for these elements.

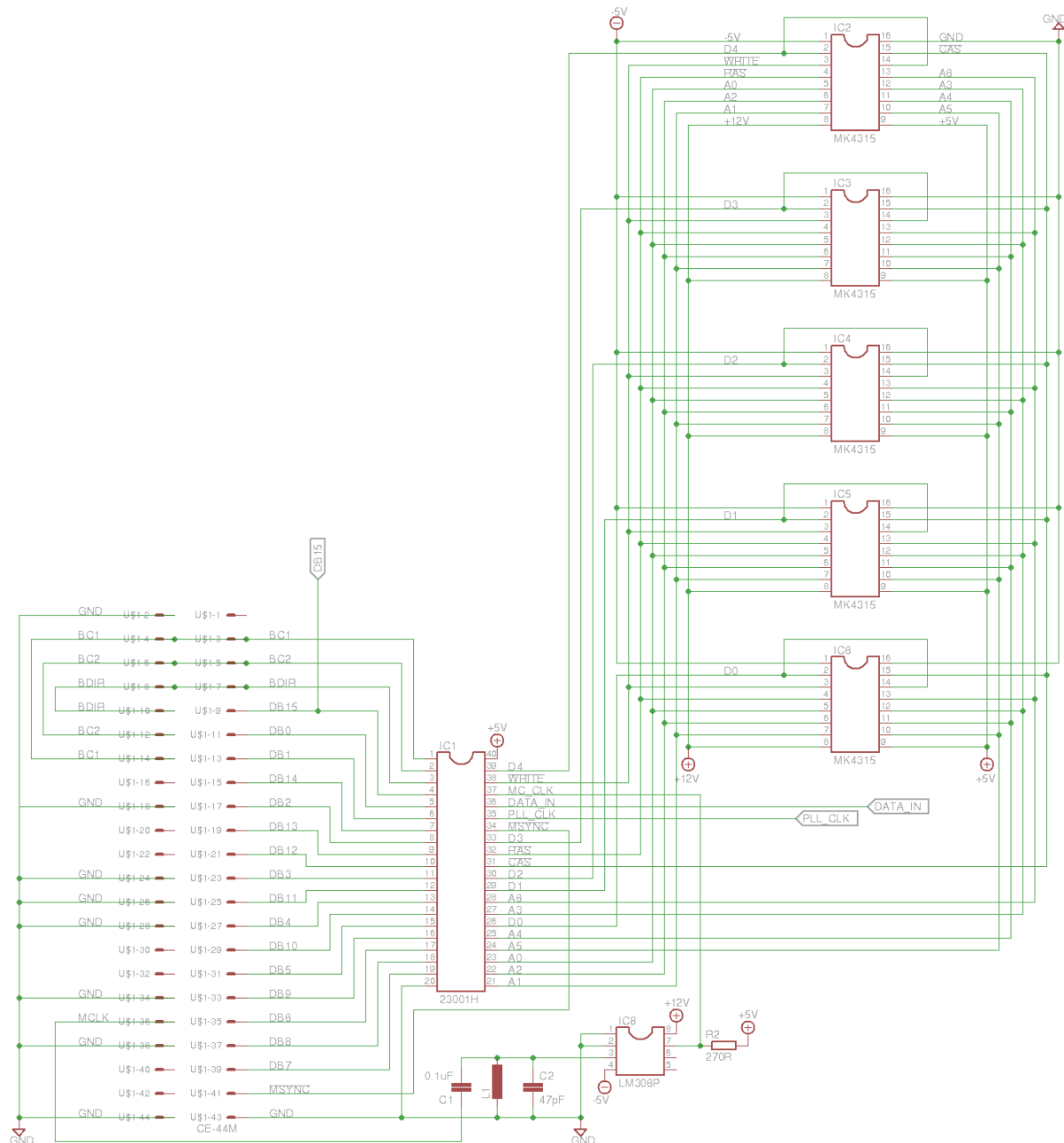


Figure 8 : PlayCable Digital Subsystem

This digital subsystem interfaces with the other elements of the PlayCable through three signals. The ASIC issues instructions to the PLL tuner to switch channels through the PLL_CLK (ASIC pin 35) and cartridge port DB15 pins. Digital data from the program stream is read by the ASIC on the DATA_IN line (ASIC pin 36).

The clock circuit takes MCLK from the Master Component cartridge interface and uses an LM306P amplifier to clean it up before passing it on to the PlayCable ASIC. At least two variations of the clock circuit have been found within General Instrument PlayCables. The variation is believed to be required to ensure compatibility with the Intellivision II as described in Section 0.

6.2 PlayCable ASIC

The brain of the PlayCable is the ASIC, part number 134-550 23001H, manufactured by General Instrument, as shown in Figure 9.

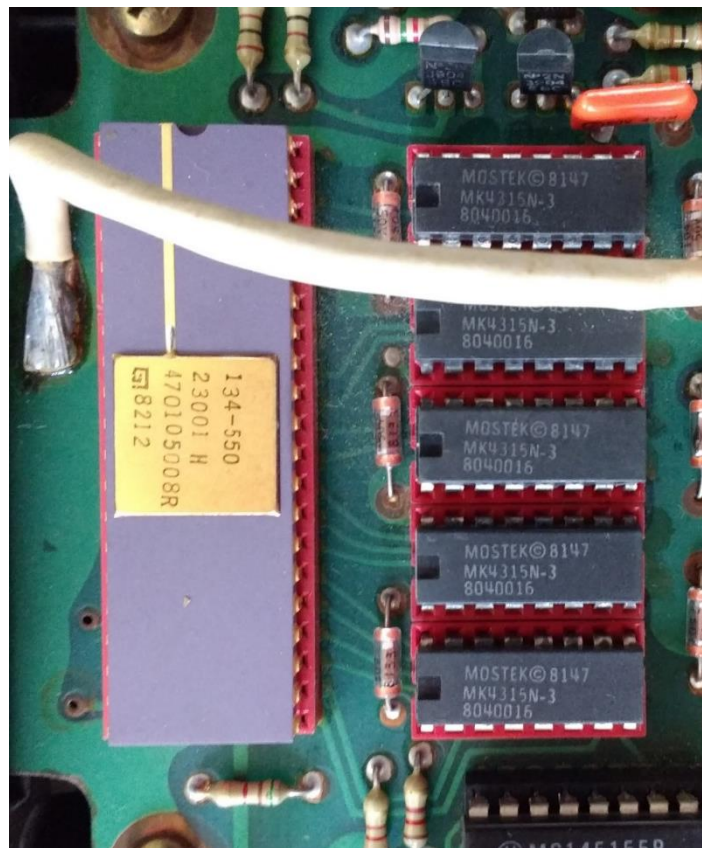


Figure 9 : PlayCable ASIC and RAM

The ASIC controls the state of the overall system and can be inferred to contain:

- An interface to the Master Component CP-1610 CPU bus
- A ROM containing the 512 decle PlayCable bootstrap / monitor / EXEC program
- An interface to the PlayCable's internal DRAM
- A module to control the PLL cable tuner
- A module to receive and interpret serial data from the cable demodulator
- A state machine to integrate and co-ordinate the various modules

The following sections examine each of these elements.

6.2.1 Master Component Interface

The ASIC must demultiplex the combined address and data bus used by the Intellivision's CP-1610 CPU into separate buses and then decode addresses that are relevant to it. It also has to handle a system wide reset state. This functionality will be similar to that found inside Intellivision cartridge ROMs or the Intellivision System RAM chip.

6.2.2 Bootstrap ROM

From a hardware perspective the PlayCable bootstrap ROM is 512 decles long and resides at \$4800 to \$49FF in the Master Component's memory map. Addresses in the ROM are not fully decoded by the PlayCable, and therefore, the content is repeated four times starting at \$4800, \$4A00, \$4C00 and \$4E00. More information on the software content of the ROM can be found in Section 7.1.

It is worth noting that the PlayCable ASIC does not operate without both the MCLK and MSYNC signals being driven correctly. Regular General Instrument ROMs do not have this requirement, and therefore, third party ROM dumpers may not implement these signals. As a consequence, some ROM dumpers may fail to extract the bootstrap ROM image from the PlayCable.

6.2.3 PlayCable RAM Interface

Much like the Intellivision System RAM chip, the PlayCable ASIC has to translate between two memory buses. This involves identifying relevant requests for locations in the Master Component's memory map, evaluating the equivalent locations within the PlayCable's memory and executing the request. This task is complicated by the use of five 16K x 1-bit RAM chips, as described in Section 6.3. In addition, because the PlayCable makes use of dynamic RAM the ASIC has to execute periodic refresh operations to prevent data loss.

6.2.4 PLL Tuner Interface Module

The Intellivision Master Component instructs the PlayCable to tune to a specific channel by writing the channel number, as a serial sequence of bits, to bit 15 of the PlayCable data register at address \$4FFF. For example, to tune to channel \$37 the Master Component would execute a sequence of 16 write instructions to \$4FFF, where the values written would encode the value \$37 (0000 0000 0011 0111) in bit 15. Within the PlayCable this information is transferred to the PLL chip by pin DB15 on the cartridge port being connected to the PLL data input line (PLL - pin 11). Clearly, this causes a problem, as most of the time DB15 is being used for things other than instructing the PLL. This issue is resolved by the PlayCable ASIC identifying when DB15 has data on it resulting from a write to address \$4FFF and pulsing its PLL_CLK line (ASIC - pin 35). The PLL_CLK signal is also fed to the PLL chip and causes it to read the data on PLL input pin 11 at this point. At other times the PLL ignores the input provided to it by the DB15 line.

6.2.5 Serial Receiver Interface Module

Perhaps the most interesting aspect of the PlayCable ASIC is the module that receives and processes serial data from the Demodulator. In principle it acts like the receiving component of a very inflexible Universal Asynchronous Receiver Transmitter (UART). 7-bit serial digital data is expected to be presented on pin 36 of the ASIC at a bit rate of 13.982KHz, using a Manchester phase encoding scheme:

https://en.wikipedia.org/wiki/Manchester_code

In the algorithm implemented data is sent least significant bit first with binary 0 being represented as 1 / 0 and binary 1 as 0 / 1. So, the 7-bit value \$56 (binary 1010110) would be represented as 10 01 01 10 01 10 01, as shown in Figure 10.

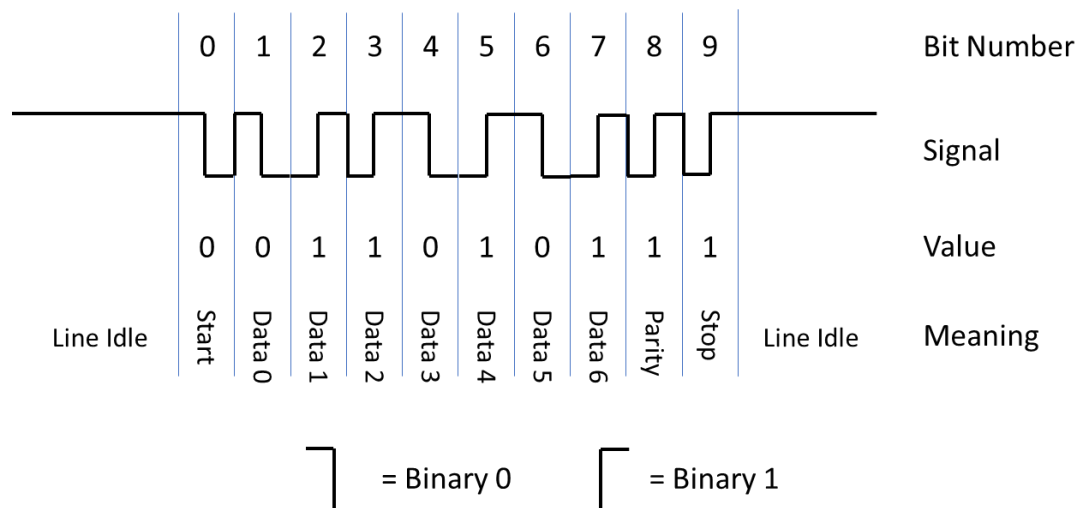


Figure 10 : PlayCable Line Transmission of the Hexadecimal Value \$56

The data bits are followed by a parity bit, such that the number of binary 1's, including the parity bit, is always odd (known as odd parity). In this example the parity bit would be a binary 1. Finally, the complete payload is surrounded by a binary 0 start bit and binary 1 stop bit to signal the transition of the line from its idle state of logic 1 to an active state. This scheme, known as 7O1 in the standard nomenclature, is similar to that used by Videotext / Teletext and EIA-608 closed caption systems.

The ASIC receives the serial data, removes the Manchester encoding and assembles the raw bit-stream into a parallel word suitable for transmission to the Master Component. On completion of a word the ASIC checks it for errors. The checks include assessing parity errors, but do not seem to include certain kinds of framing errors. For example, it is possible to insert long periods of line inactivity (the digital input being held at a high logic state) into the middle of a character without generating errors.

The current state of the receiver can be interrogated by the Master Component at any time by reading the status register at address \$4FFE. The full meaning of the data in this register is not known. However, it is known that bit 0 being high indicates that a word has been received and bits 1-3 being high are indicative of errors. The subroutines within the PlayCable bootstrap ROM treat all errors as cause to restart the loading process. The status register is reset by reading it, allowing the Master Component to indicate to the PlayCable that it has responded to new data or errors.

The data from the stream is presented to the Master Component at address \$4FFF. Only the 7 data bits of the current word are made available, the parity, start and stop bits having been stripped off. It seems unlikely that the PlayCable incorporates any kind of data buffer to prevent data loss should the Master Component fail to keep up when reading from the stream. No error detection or correction scheme, other than the single parity bit, exists at this level of the transmission protocol.

6.3 RAM

The PlayCable contains 16K x 5 bits of RAM spread across either five MK4315 or MC4116 16K x 1-bit dynamic RAM chips. Clearly, the Master Component requires either 10-bit or 16-bit memory for program storage, therefore, the PlayCable ASIC must map its 5-bit wide memory into 8K x 10 bits. This implies that whenever the Master Component reads from an address in PlayCable memory space, the ASIC is quickly retrieving data from two different locations in PlayCable RAM and concatenating the two 5-bit nickles it finds together, before returning the resulting decle. The

opposite will occur on memory writes, with the ASIC splitting a decele received from the Master Component into two 5-bit nickles, before writing them to two different locations in its memory.

Because the chips used are dynamic RAM, based on capacitors, rather than the more expensive, transistor based static RAM used in the Master Component, they require a periodic refresh operation to prevent the loss of data. Managing this process is also the responsibility of the PlayCable ASIC.

6.4 RF Cable Receiver

The RF cable receiver is located on a daughterboard within an RF shield at the top of the PlayCable. The RF shield is marked with the part number 142-110-00 with a date code of 8130, as shown in Figure 11.

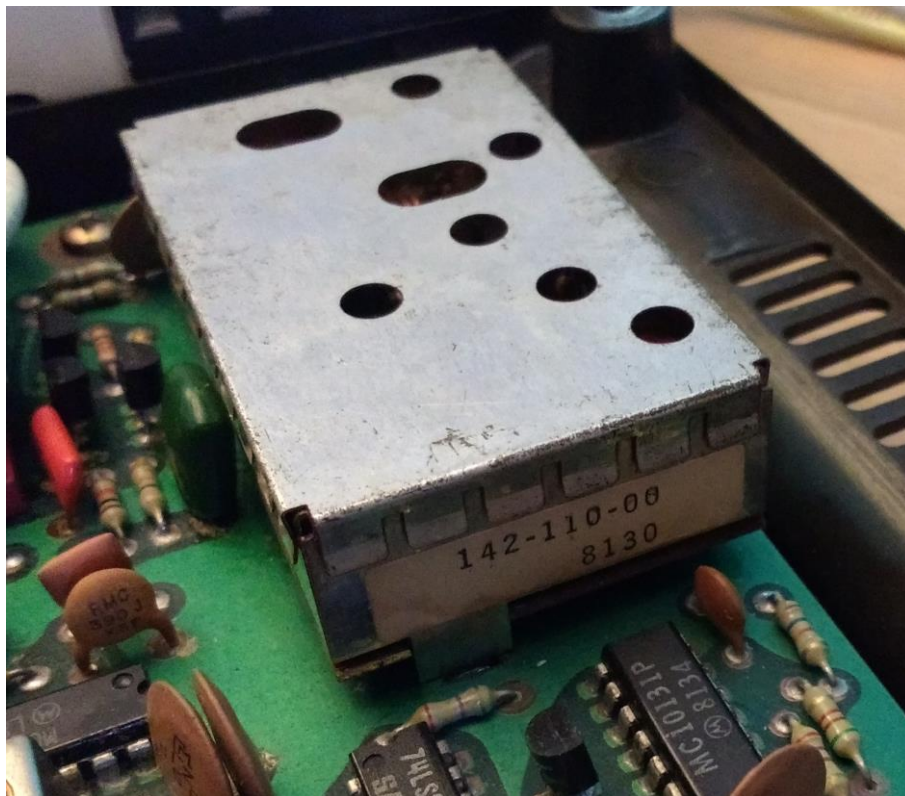


Figure 11 : PlayCable RF Receiver

Interestingly, unlike most RF modulators, the cable socket is not integrated into the RF receiver, but is mounted to the main PlayCable PCB (bottom left of Figure 6). The RF receiver has not been opened and the components within it have not been investigated. No information has been found on the Internet regarding this component. However, from the behaviour of both the PLL tuner and demodulator circuits it has been possible to work out that the RF receiver is a [heterodyne](#) tuner that uses high-side injection of its local oscillator and emits a 10.7MHz signal for onward processing.

6.5 PLL Tuner

The tuner is responsible for controlling which of the game channels the RF receiver is tuned to. It is based around an MC145155 Phase Locked Loop (PLL) chip. This monitors an externally generated signal and assesses whether it has a desired frequency. It then generates signals that indicate whether the source frequency should be increased or decreased to match the required frequency.

The overall mechanism is very similar to that used by a digital FM radio receiver. The circuit schematic for the PLL tuner is shown in Figure 12.

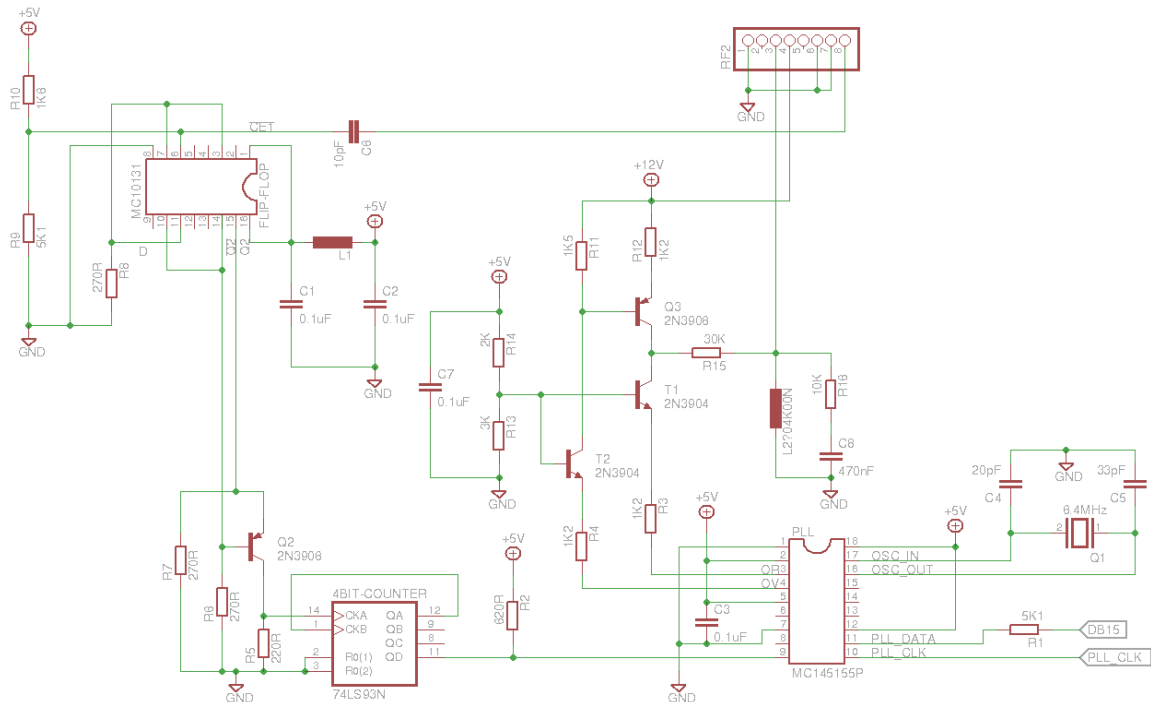


Figure 12 : PlayCable Tuner Schematic

As described in Section 6.2.4, the ASIC configures the PLL circuit by sending a sequence of 16 pulses from AD15 to the PLL_DATA input on pin 11. This data is clocked into the PLL using the PLL_CLK signal fed into pin 10. This configuration sets a divisor value within the PLL chip.

A carrier frequency input is received from pin 8 of the RF receiver's RF2 socket into pin 6 of an MC10131 D-type flip flop. The MC10131 divides the frequency by 4 before passing it on to a 74LS93 4-bit counter that further divides it by 16. The resulting divided by 64 signal is then passed into the input of the MC145155 on pin 9.

The PLL chip compares this input frequency with one derived from its 6.4MHz clock. The comparison is made by dividing the input frequency by the configuration value provided by the ASIC. The result is then compared with 1562.5Hz. This value is constructed by dividing the 6.4MHz clock by 4096 (this divisor is configured by the voltages applied to pins 1, 2 and 18 of the PLL). Depending upon whether the sensed carrier frequency, once divided by this large number, is above or below the reference value one of ϕ_R or ϕ_V (pins 3 and 4) will be pulsed. It is known that the bootstrap ROM is hardcoded to configure the PLL with the value \$4A0 when configuring it to load the menu program. As a consequence, it can be calculated that the reference frequency being tuned to is 118.4MHz (Reference frequency: 1562.5Hz x PLL configuration: \$4A0 or 1184 x External pre-scale: 64). Further, the PLL can attempt to tune to any frequency between 300KHz and 1.64GHz in steps of 100KHz. However, this theoretical range will be limited by the capabilities of the RF receiver. It should be noted that this is the frequency that the local oscillator of the RF receiver is set to, not the frequency on which the menu program is being broadcast.

The phase error signals ϕ_R and ϕ_V that are generated by the PLL are combined by the three transistors T1, T2 and Q3 resulting in a signal that swings between 0V and 12V if ϕ_R or ϕ_V are pulsed low respectively. These pulses are integrated by the passive components connected to the RF

receiver on RF2 pin 3, giving a voltage between 0V and 12V. This signal is used to set the carrier frequency within the receiver, tuning it and creating a closed loop control system.

6.6 Demodulator

The demodulator is the least understood component of the PlayCable, mainly owing to my rusty analogue electronics knowledge. Therefore, the following description should be viewed with caution. If others can validate and expand on this work it would be appreciated. The demodulator takes a filtered channel signal from the RF receiver and extracts a digital game stream from it. In doing so it generates the Manchester phase encoded digital stream required by the PlayCable ASIC. The schematic of the demodulator is shown in Figure 13.

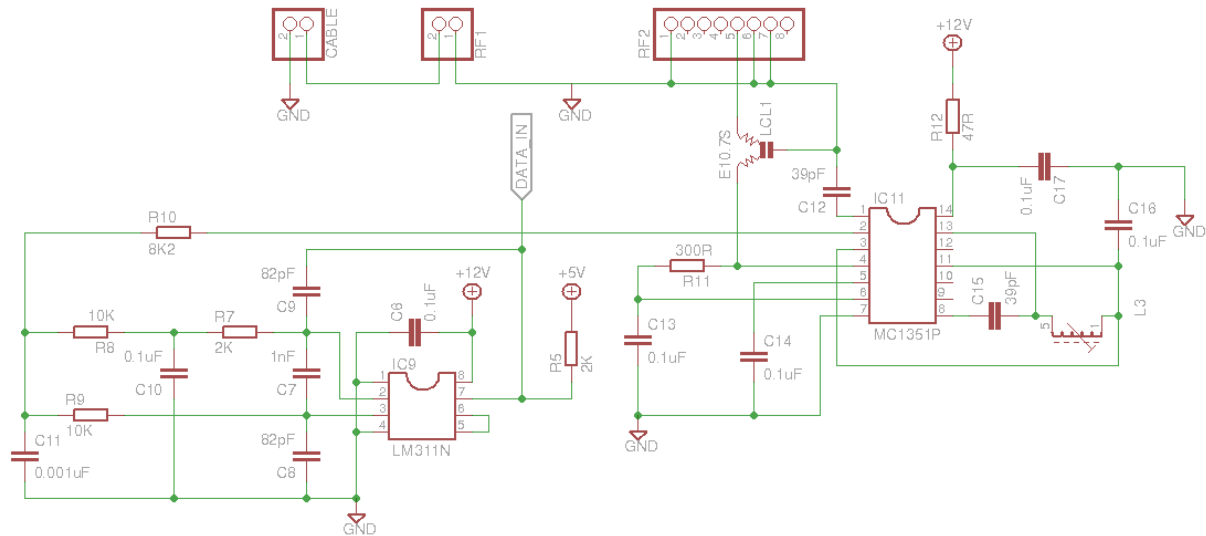


Figure 13 : PlayCable Demodulator Schematic

It is believed that the RF receiver passes an FM encoded audio signal to pin 4 of the MC1351P FM receiver circuit. This chip is more normally used in a TV or radio receiver to extract and decode audio from the broadcast signal.

The signal fed to the MC1351P is encoded onto a 10.7MHz carrier. While this might seem a rather strange frequency, it is one of the [commonly used intermediate frequencies](#) used when decoding FM radio transmissions; something that again highlights the close relationship between FM radio and the PlayCable. The use of a 10.7MHz intermediate frequency resolves a discrepancy in the menu frequency between the system description provided by Dages and the analysis of the PLL tuner circuit in section 6.4. Dages suggests that the directory or menu frequency is 107.7MHz, as shown in Figure 14, however, the configuration of the local oscillator is set to 118.4MHz.

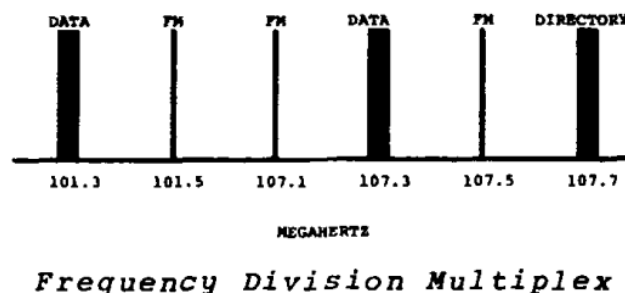


Figure 14 : PlayCable Signal Frequencies (Dages)

By tuning the local oscillator to a frequency 10.7MHz higher than that of the actual menu channel the high-side injected [heterodyne](#) circuit within the RF receiver will emit a signal at 10.7MHz with the data still encoded in it. The MC1351P can then be configured as a regular FM radio receiver to extract an audio frequency signal from this 10.7MHz carrier. Therefore, tuning the local oscillator to 118.4MHz is consistent with the menu program being broadcast at 107.7MHz.

The MC1351P emits an “audio” signal containing the data stream from pin 2. If this signal were to be played back through a hi-fi it would sound similar to a very high pitch computer audio tape from the early 1980s. The audio is forwarded to what is understood to be a band pass filter. This will identify signals close to the operating frequency of 13.98KHz, pulling the DATA_IN line to ground when they are found. For all other frequencies the resistor R5 pulls the DATA_IN line to 5V. As a consequence, short bursts of the appropriate frequency can be used to generate a digital input to the ASIC.

6.7 Power Supply

The final part of the PlayCable hardware is its internal power supply. This takes a standard 110-115VAC 60Hz mains supply and generates the necessary DC output required to run the PlayCable. It does this by stepping the incoming mains supply down to two inputs of 15VAC and 20VAC with a multi-tap transformer.

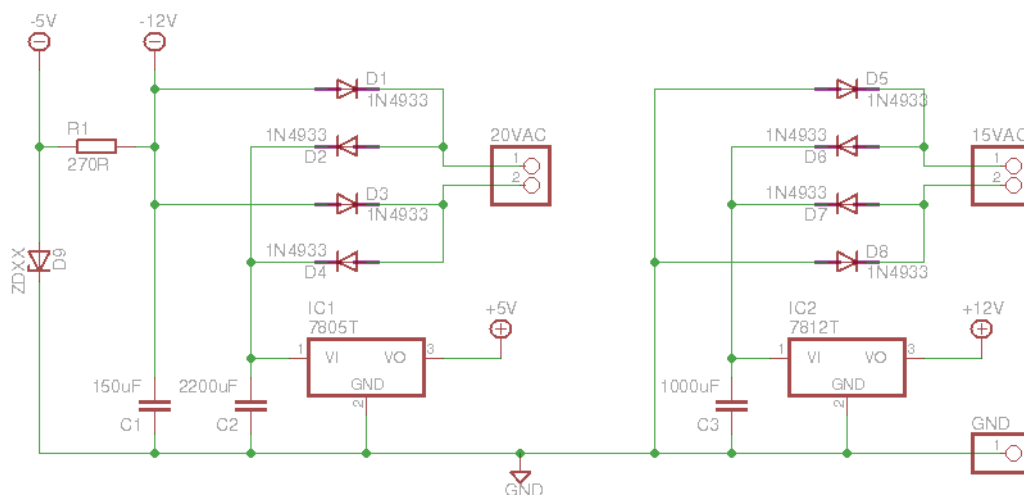


Figure 15 : PlayCable Power Supply Schematic

The 15VAC source is used to create a regulated 12VDC supply. The 20VAC source is used to create a regulated 5VDC supply and unregulated -5VDC and -12VDC supplies. The power supply is fairly standard, as can be seen in Figure 15. Despite having a heatsink the 5VDC regulator seems to get rather hot.

7 Software

The software associated with the PlayCable adapter comes in three parts. The bootstrap or EXEC ROM built into the adapter itself, a menu program streamed to the adapter on a known channel at start up and finally game ROMs or other programs subsequently chosen by the user using the menu.

7.1 PlayCable Bootstrap ROM

The bootstrap ROM is 512 decles long, its primary function is to initialise the PlayCable to a predefined channel, display the PlayCable title screen, then search, load and run a program named "DIR1 CATALOG". Having done this all control passes to the program loaded.

In completing this task, the bootstrap ROM defines the data format used to distribute content. It is clear that in order to send 10-bit Intellivision data it must be split across the 7-bit characters sent in the data stream and reconstituted in the PlayCable adapter. It is also clear that some form of signalling is required to indicate when a stream starts and ends. To fulfil these functions the PlayCable uses a rather convoluted data format built on top of the 7-bit character protocol described in Section 6.2.5.

Working from the bottom up. The value \$7F is used as a special delimiter character. All other values should be viewed as either a 1-bit type field in the MSB with a 6-bit payload or a 2-bit type field with a 5-bit payload. Details of the character formats are provided in Figure 16. Data sent down the wire can either be a 16-bit value, used to change the target address of subsequent data writes, or a 10-bit value to be written to RAM.

Format	Meaning
1111111	delimiter
1xxxxxx	Bits 10-15 of a 16-bit address (note that addresses of \$FC00 and above cannot be set owing to a clash with the delimiter)
01xxxxx	Bits 5-9 of a 16-bit address
00xxxxx	Bits 5-9 of 10-bit data or bits 0-4 of either a 16-bit address or 10-bit data

Figure 16 : PlayCable Stream Character Format

All data is sent most significant bits first. For example, the 16-bit address \$5123 would be sent as \$54 (type 1 with the payload 010100), followed by \$29 (type 01 with the payload 01001), followed by \$03 (type 00 with the payload 00011). Similarly, the 10-bit data value \$234 would be sent as \$11 (type 00 with the payload 10001) followed by \$14 (type 00 with the payload 10100).

The structure of the overall stream is then composed of a header, containing the name of the program, an instruction to set the starting address to write to, and the program data itself. The header and program data are encoded as 10-bit data values; with the address sent as a 16-bit value. Each section of the stream is terminated with a \$7F delimiter and the end of the program is marked with a sequence of nine or more \$7F delimiters. The overall stream structure is summarised in Figure 17.

Field	Structure	Notes
Header	DIR1 CATALOG	The ASCII name of the program encoded as 10-bit data values, as described above
Checksum	\$B5	The value necessary to make the sum of the header data a multiple of \$400, encoded as a 10-bit data value
End of Header	\$7F	Marks the end of the header, forces validation of the checksum, prior to loading the game
Address Set	\$54 \$20 \$00	Set the data load address to \$5000 using a 16-bit address encoded value. Without this the bootstrap will attempt to overwrite itself!
End of Address	\$7F	Latch address value ready to start program load
Program Data	...	Program data sent as 10-bit encoded data pairs
Checksum	...	Value necessary to make the sum of the program data a multiple of \$400
End of Program	9x \$7F	Nine delimiter characters are treated as the end of program marker. The first terminator is used to force the checksum validation and game start. The remaining delimiters are used when searching for a title to locate the start of the next program and initiate the name match

Figure 17 : PlayCable Stream Format

As the PlayCable EXEC ROM starts at address \$4800 it takes over from the Master Component EXEC fairly early, before much of the Intellivision bootstrap has been completed. Once the PlayCable EXEC has control the bootstrap process is as follows.

- A default command located at \$4802 is set up in R4. An instruction contains the channel to tune to, a decle containing loading options and the name of the program, in this case "DIR1 CATALOG".
- \$4815 – Believed to be the intended main entry point for the menu program to request programs to be loaded.
- \$481B - The PlayCable tunes to the correct channel, \$4A0 in the case of the menu, by writing this value as a sequence of bits to bit 15 of \$4FFF
- \$4831 - The loading options decle is read and stored
- \$4843 - A one-shot Interrupt Service Routine is set up and the boot pauses
- \$49CD - When the ISR fires at the next video frame, the STIC is initialised and the standard Mattel title screen is drawn.
- \$4846 - On completion of the one-shot ISR, the main bootstrap restarts
- \$4849 - The PlayCable name and Jerrold copyright message are written to the screen
- \$4875 - The second part of the game title, starting from the 7th character (e.g. "CATALOG") is written to the screen as the game title. This leads to the loading screen looking like Figure 18. Note the eight coloured bars at the top of the image.

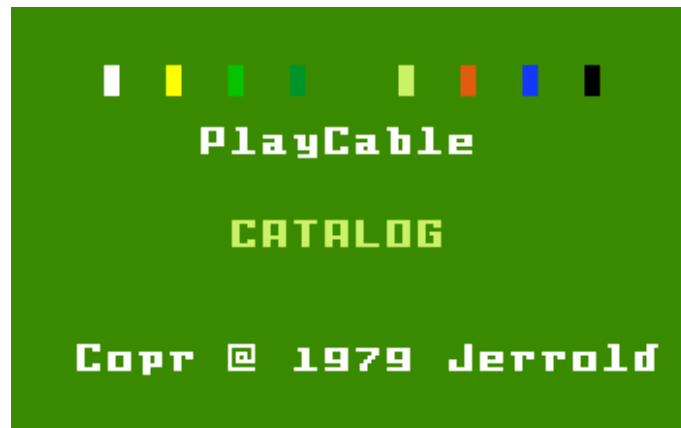


Figure 18 : PlayCable Loading Screen

- \$487C - A short ding is sounded to indicate the start of the search phase
- \$488B – Wait for the end of the current program, as marked by nine or more \$7F terminators. Sound a short ding every 1K bytes of data read from the wire.
- \$4897 – Compare the program name on the wire with the full name of the program being sought. Keep going until a \$7F terminator is received, at which point go to \$48A1. If an incorrect character is found, go back to \$488B and start looking for the end of the program.
- \$48A1 – Validate that the checksum, which is a simple sum of all the bytes read since the start of the program, is a multiple of \$400 (i.e. the 10 LSBs are 0). If it isn't, go back to \$488B and start looking for the end of the program.
- \$48AA – Read to the next non-delimiter character. This section is believed to be used to set R4 to the correct value so that subsequent writes occur at the correct address. Typically, this value will be \$5000. Because of the way in which the data protocol works this can be encoded as \$54, \$20, \$00 followed by a \$7f delimiter to mark the end of the section.
- \$48B1 – At this point the bootstrap switches into program reading mode. 10-bit data is read from the wire and written to the address held in R4. This continues until a \$7f delimiter character is received. As the program is loaded the number of decles received is counted. Depending on the value of the loading options, a long ding may be sounded every 1024 decles and one of the coloured bars at the top of the loading screen is turned white, as shown in Figure 19. As a consequence, there is both a visual and auditory indication of the size of the program being loaded, in this case the 6K game Sub Hunt. When a \$7f character is received control loops back to \$48A1 for the checksum to be tested.



Figure 19 : PlayCable Sub Hunt Loading Screen

- \$48AE – Assuming the checksum is passed, the stream is read until the first non-\$7F delimiter is found. If this results in more than nine \$7F delimiters being found the program is assumed to be complete and the boot process will start.
- \$48CD – The loading options specified at the start of the process, as well as the “key clicks” cartridge header entry in the game, now affect how the game will load. It is possible for the bootstrap to show an interstitial “READY” screen, as shown in Figure 20, and play a short fanfare.



Figure 20 : PlayCable Interstitial “READY” Screen

- The name of the game is read from the cartridge header and written to the screen. The disc must then be tapped to start the game and bring up its title screen. In practice, judging by dave1dmarx’s recordings these features were not used in practice and the game started immediately on the load completing.

Beyond the address \$491E the bootstrap ROM contains subroutines used by the main bootstrap process. These include the code for reading decles and 16-bit addresses from the wire; and processing incoming data including emitting dings and colouring title blocks on screen.

There are some interesting aspects to this mechanism. If the ASIC signals an error in the status register the bootstrap ROM will emit a long ding and restart the loading process. As a consequence, if a string of errors occurs, the Intellivision will make a continuous squeal:

<https://www.youtube.com/watch?v=E70MR-E5Pgk>

If the cable feed is disconnected from the PlayCable it appears that the RF receiver will generate noise on the digital input to the ASIC, causing it to report just such a sequence of errors and leading to the annoying squeal continuing indefinitely.

There is no way to bypass this bootstrap process, even if a game has already been loaded into memory. Therefore, if a game does not provide a means of restarting without pressing reset on the Master Component the menu will have to be reloaded. As a consequence, it will be necessary to find, select and reload the game again to play a second time.

There is some circumstantial evidence that multiple games were transmitted on a single game channel in 1983. This is based on a contemporary audio recording of Buzz Bombers being loaded, provided by dave1dmarx:

<http://atariage.com/forums/topic/284228-playcable-research-and-development/page-3#entry4166618>

On this recording, prior to the game being found there would seem to be a period of searching that is too large to be a single instance of Buzz Bombers. Within this period a “double ding” is heard, suggesting a game header has been found that does not match that expected and the search is being resumed. After this double ding 4K decodes of data are read from the stream before the start of Buzz Bombers is found. However, Buzz Bombers is an 8K game, and therefore, the 4K decodes read is too small to represent a copy of it. This suggests that the 8K game Buzz Bombers might have been paired with at least one 4K program on the same stream.

By setting up the channel number, loading options and game title structure in memory it is possible for a program running in PlayCable RAM to request another game to be loaded by calling into the bootstrap ROM. Requests can be made to load games with the PlayCable title screen being drawn, dings emitted, etc, as is done by the menu program. Once the game title is matched, the memory structure containing the loading instruction is not used again and can be overwritten by game code. As a consequence, the full 8K of PlayCable RAM can be used for a game, even if the menu program and loading request is found there.

The structure of the PlayCable EXEC also makes the construction of large, multi-section programs possible. Loading requests can be constructed that don’t draw the title screen or emit dings. By choosing an appropriate target address it would be possible to “page in” levels or sections of gameplay in the middle of a program. It may be possible to play music or animate the screen whilst doing this using a non-standard ISR. However, some care would be needed to ensure that this did not consume too much CPU time and cause data loss when reading the stream.

As has been noted, program names come in two parts (e.g. “DIR1 CATALOG”). The second section, starting at the 7th character, can vary in length and is shown to the user as the name of the game. The first part is exactly 6 characters long, used for matching and is never seen by users. It is not known whether this field had any other purpose. However, it might not be a coincidence that file names within RT-11 (one of the common operating systems used by PDP-11 computers) are six characters in length.

7.2 PlayCable Menu Program

The menu program used to select games on the PlayCable is not contained within the bootstrap ROM, but was streamed on the predefined channel \$4A0, equivalent to 107.7MHz. This meant that the contents of the program might have differed by cable company. It is known that the menu program was changed over time. As far as is known, all versions of this program are lost to history.

The program comprised a splash screen, as shown in Figure 21, followed by a number of menu screens, each with four game choices.



Figure 21 : PlayCable Splash Screen (http://thedoteaters.com/?attachment_id=6657)

Figure 22 shows an alternative splash screen containing the more rounded representation of the PlayCable logo that was registered by Jerrold in late 1978.



Figure 22 : Jerrold PlayCable Splash Screen (<http://www.intvfunhouse.com/hardware/playcable/>)

Once the menu program loaded, the splash screen was displayed automatically and a tune started to play. The tune would play to completion once, even if the user started to browse the games available. Recordings of these tunes were made by dave1dmarx and subsequently uploaded to YouTube. In late 1982 one of three songs could be played “at random”. It is hypothesised that in reality three different versions of the menu program were circulating on the stream one after another, each with a different tune embedded in it. Which tune played would depend upon when the user hit reset to start the menu program search and which of the menu programs happened to be the next on the stream. Using this approach would minimise the download time of each menu, as no redundant tune data would be loaded. At this point the tunes available were:

- The Entertainer - https://www.youtube.com/watch?v=WvZp_k6s33o
- Music Box Dancer - <https://www.youtube.com/watch?v=0CX0vEg-WE>
- 12th Street Rag - <https://www.youtube.com/watch?v=IDfvzo4vAsA>

It is interesting to note that the tunes are very different lengths although this does not seem to have had a significant impact on the loading time of the menu program. Strange sounds can also be heard at the end of the recording of the Entertainer. This is believed to be the result of a bug in the program, probably the omission of an “end of music” marker. As a consequence, the player program is believed to be continuing, interpreting the subsequent menu program and data as musical notes.

dave1dmarx also made some additional recordings in early 1983. At this point only one variant of the menu program existed which played The Entertainer. The bug found in the original version was fixed in the interim and beeps at the end of the recording are no longer apparent.

At any point after the splash screen has been displayed the user can tap the disc and a sequence of menu screens will be shown. No images of the original menu screens are known to exist, however, based on the recollections of dave1dmarx, the replica shown in Figure 23 has been constructed.



Figure 23 : Replica Menu Program Running Within JzIntv

Games are believed to have been listed in alphabetic order, four to a page. Games were selected by choosing the game number and pressing ENTER on the controller keypad. Pressing the disc advanced the menu to the next page. Pressing the disc on the final page resulted in the menu cycling and the splash screen being shown again.

It is not known whether the catalogue of games available each month was baked into the menu program, as has been done with this replica, or was subsequently loaded from either the same or a different channel to the main body of the menu. Separating the catalogue from the menu program and using the “paging in” mechanism described above might have made the incremental monthly game switchover described in Section 0 simpler to implement.

It also would have been technically possible for the menu program to sample channels to see whether they were active and data was being transmitted. This information could then have been used to remove channels from the menu that were inactive. Again, it is not known if this was done.

On choosing a game to play it is believed that the menu would set up a request comprising the channel, loading options and game name. It would then relinquish control, jumping into the bootstrap ROM which would execute the request and load the game.

7.3 PlayCable Game ROMs

The games downloaded via the PlayCable service are believed to be unchanged from the versions sold on cartridges in stores. It is known that the architecture of the PlayCable adapter software means that the full 8K of PlayCable RAM is available for games. In addition to the 6K games found in the game roster from 1983, as seen in Figure 24, dave1dmarx has an audio recording of the 8K game Buzz Bombers loading and playing as described in Section 7.2.

PlayCable™

THE ALL GAME CHANNEL

Game Plan

January–April, 1983

	LIBRARY	JANUARY LINE-UP	FEBRUARY LINE-UP	MARCH LINE-UP	APRIL LINE-UP	RETURN DATE
SPACE NETWORK	Astroblast™					June
	Space Armada™					
	Space Battle™					
	Space Hawk™					
	Star Strike™					June
SPORTS NETWORK	Auto Racing					May
	Baseball					
	Basketball					
	Bowling					
	Boxing					
	Football					
	Golf					
	Hockey					August
	Skiing					August
	Soccer					
	Tennis					May
	Horse Racing					
GAMING NETWORK	Poker/Blackjack		NEW!!			August
	Roulette					June
	Royal Dealer™	NEW!!				July
	Advanced Dungeons & Dragons™					May
	Armor Battle®					June
ACTION NETWORK	Frog Bog™					May
	Lock 'N' Chase™					
	Night Stalker™					May
	Sea Battle™					
	Shark! Shark!™				NEW!!	
	Sharp Shot™			NEW!!		
	Snafu™					
	Sub Hunt™					June
STRATEGY NETWORK	Triple Action™					
	Backgammon					May
	Checkers					
	Reversi™					July
LEARNING NETWORK	Utopia™					
	Math Fun™					May
	Word Fun™					

☒ ON
☐ OFF

Figure 24 : Spring 1983 PlayCable Game Roster (<http://atariage.com/forums/topic/203583-intellivision-playcable/#entry2822809>)

Therefore, the requirements for a game to be compatible with the PlayCable are relatively simple. It must be 8K or less in size and use a standard memory map from \$5000 - \$6FFF to work without alteration.

Of the 61 Intellivision games released by Mattel, 13 are known to be incompatible with the PlayCable, as shown in Figure 25.

Game	Year	Reason
B-17 Bomber	1982	12k / Intellivoice
Bomb Squad	1982	12k / Intellivoice
Space Spartans	1982	Intellivoice
USCF Chess	1982	RAM at \$D000
Bump 'n' Jump	1983	16k
Masters of the Universe: The Power of He-Man	1983	16K
Melody Blaster	1983	12K
Mind Strike	1983	12K
Mr. Basic Meets Bits 'N Bytes	1983	12K
Pinball	1983	12K
The Jetsons' Ways With Words	1983	12K
Tron: Solar Sailer	1983	12k / Intellivoice
Vectron	1983	12k
World Series Major League Baseball	1983	24K / Intellivoice

Figure 25 : List of Mattel Games not Compatible with the PlayCable

With the exception of UCSF Chess which has a block of RAM at \$D000 the other games are all too large to fit into the PlayCable's 8K of memory. Space Spartans, the only 8K Intellivoice game is partially compatible and will run, however, no voice is heard apart from "Mattel Electronics Presents" on the title screen. The reason for this is that the Intellivoice cartridge signals are tied to ground within the PlayCable. Scooby Doo's Maze Chase, the single 8K ECS title, runs without problems. From the introduction of the Intellivision through to 1982 all games produced could run on the PlayCable. The four incompatible games released in 1982 reduced the compatibility rate to 90%. This fell to 79% in 1983 as larger games came to market.

It should also be noted that the length of games has a significant influence on the performance of the PlayCable service. As described in Section 7.1, if the Intellivision reset button is pressed the PlayCable will restart and reload the menu program from the stream and there is no way to bypass this behaviour. Therefore, if a game does not provide a mechanism to start a fresh game without pressing reset (for example, in the way that the numeric keypad does in Sharp Shot) it is necessary to cycle through the menu program between each game. Loading the menu takes 3 seconds on average, and having selected a 4K game, loading it takes a further 9 seconds on average. This game loading time rises to between 9 and 18 seconds for a 6K game; and an 8K game would take between 12 and 23 seconds. This means that it would take between 20 and 30 seconds to restart play of an 8K game. Whilst this is small relative to the 3 to 5 minutes required to load a home computer game from cassette, the fact that it could be repeated between each play of a game is likely to be frustrating to the user.

8 PlayCable Adapter Variations

The following sections describe various known variations and adaptations of the PlayCable adapter.

8.1 Jerrold PlayCable

As already noted, the Jerrold branded PlayCable is believed to pre-date the General Instrument version. This view is based on the relative filing dates of the two PlayCable trademarks, as seen in Figure 26.



Figure 26 : PlayCable Trademark Registrations (<http://tess2.uspto.gov/>)

AtariAge user intvsteve has kindly provided high resolution images of both the outside and internals of his Jerrold branded PlayCable.



Figure 27 : Jerrold PlayCable Exterior

It is immediately apparent that internally the Jerrold PlayCable is very different to the General Instrument version. The electronics are split across two large PCBs that are stacked on top of each other, as shown in Figure 28. The two boards are linked by a ribbon cable and are separated by a metal standoff, that sits beneath the entirety of the upper board. This PlayCable is also clearly an early unit with all integrated circuits having date codes of 1978 or 1979. This contrasts with the General Instrument versions where the ASICs for example are dated early in 1982.

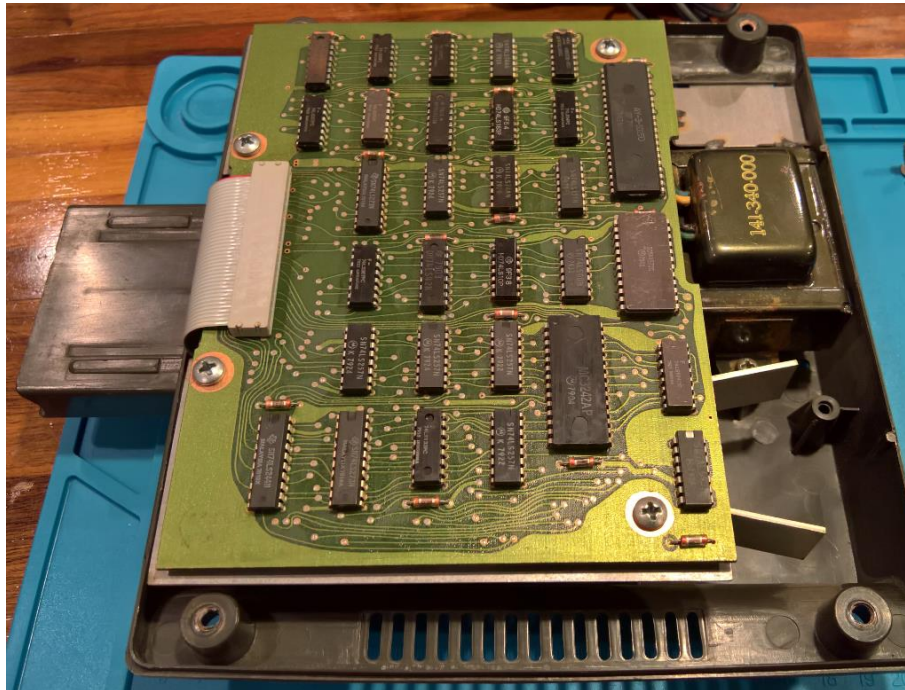


Figure 28 : Jerrold PlayCable Interior

The lower PCB is very similar to the General Instrument version, as shown in Figure 29.

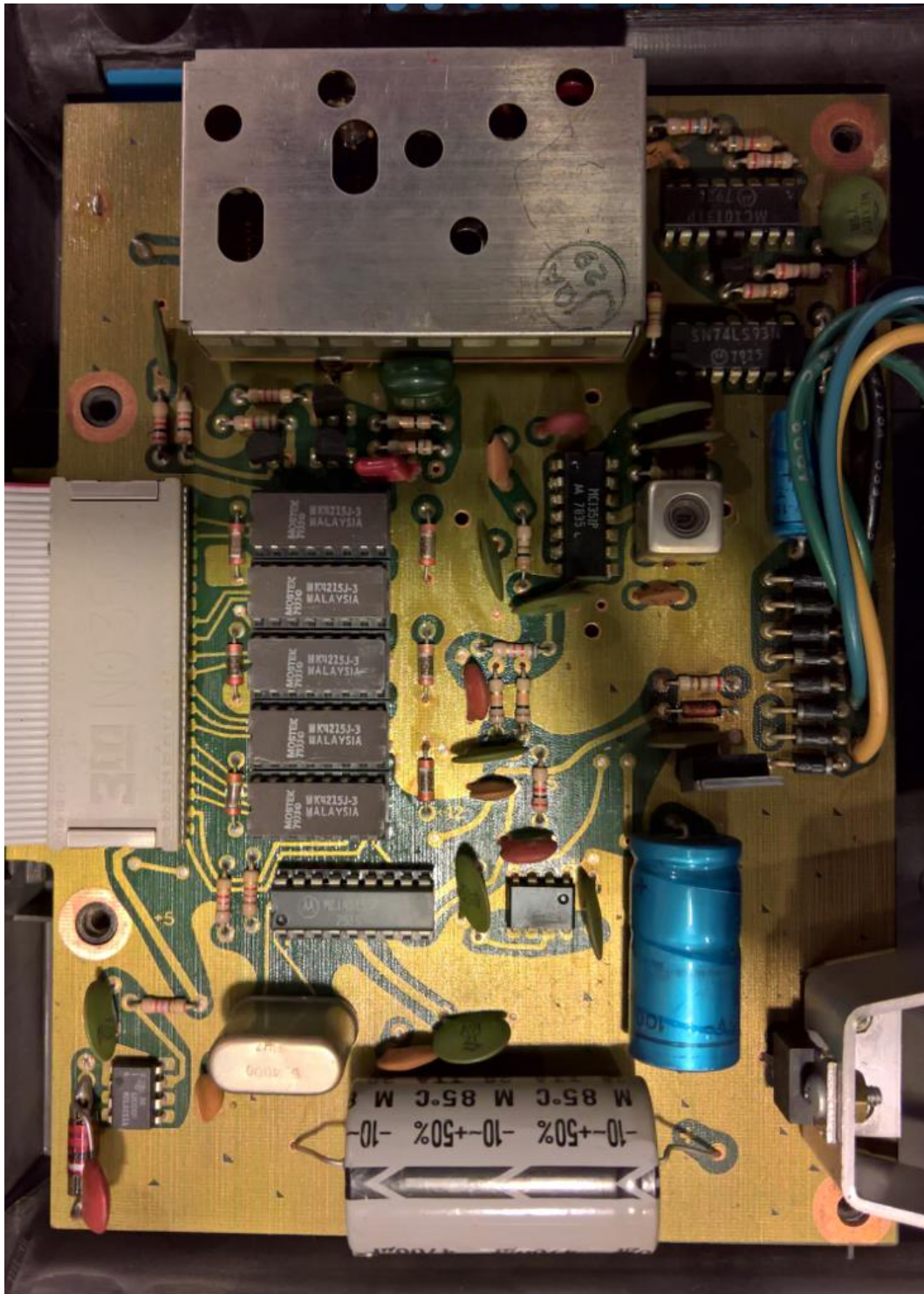


Figure 29 : Jerrold PlayCable Lower PCB Component Side

In addition to the one obvious difference, that the ASIC is replaced with a ribbon cable, a number of smaller changes are also apparent. The white earth bonding wire that traverses the GI board is not present and neither is the 5K1 current limiting resistor that links DB15 to pin 11 of the PLL chip. As a consequence, the board has a different revision number to that found in the General Instrument PlayCable, in this case REV 1.

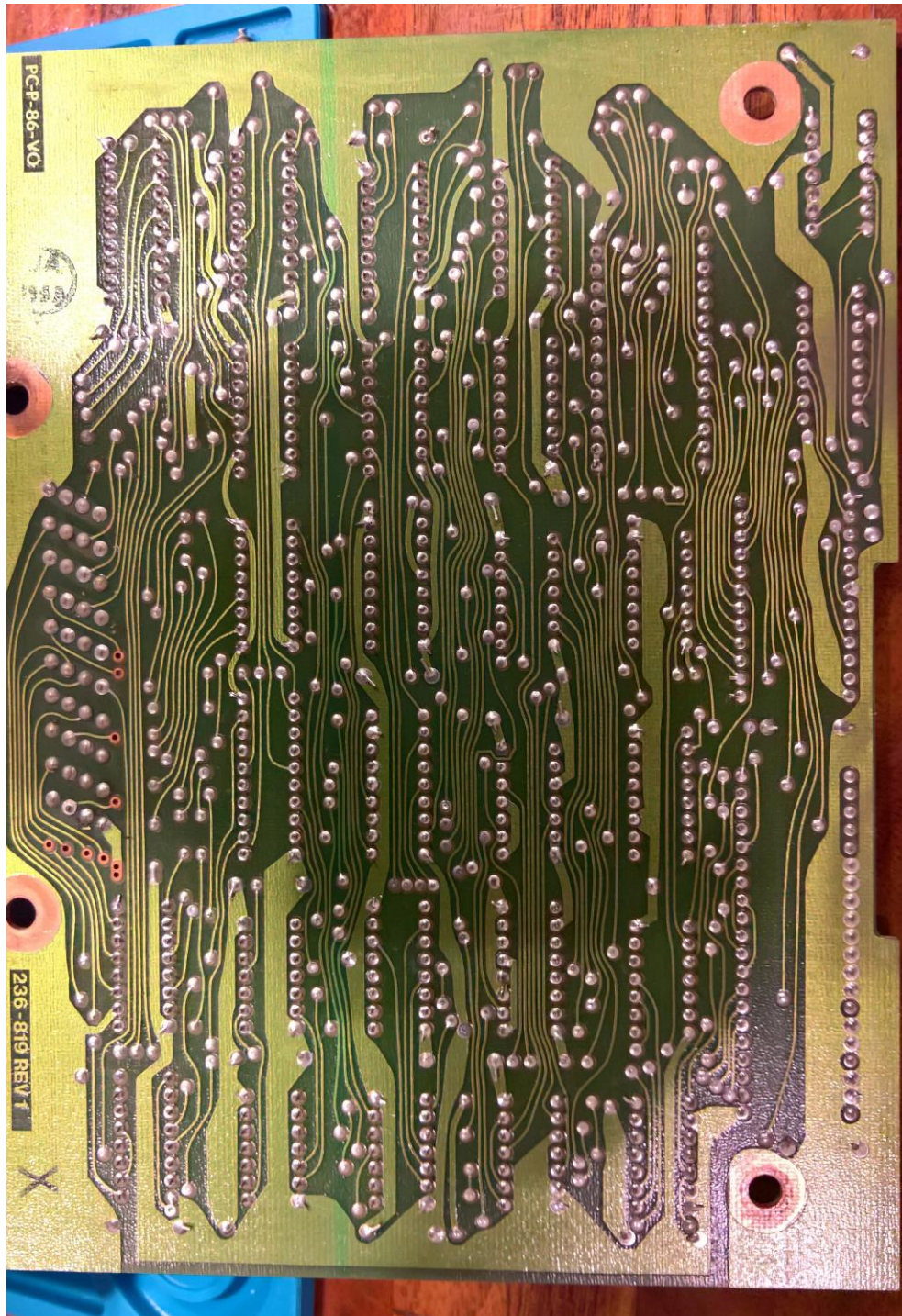


Figure 31 : Jerrold PlayCable Upper PCB Solder Side

It is possible to identify several of the key functions of the ASIC and allocate them to the larger integrated circuits on the board, as shown in Figure 32.

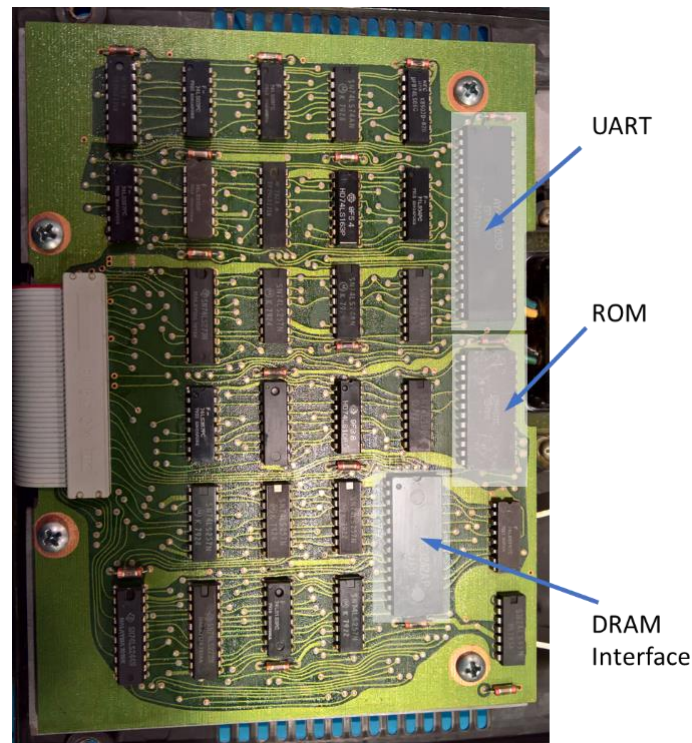


Figure 32 : Jerrold PlayCable Upper PCB Major Components

There are a number of very interesting points about this board. Firstly, and most importantly, all the components have standard pinouts and most are standard components. The UART is a General Instrument AY-3-1015, the ROM a Motorola SCM44572 and the DRAM interface support chip a Motorola MC3242. The remaining integrated circuits are standard 74 series logic and will provide the CP-1610 bus interface, PLL tuner interface, Manchester decoder and controller state machine. The use of standard components suggests that this design might be closely related to the hardware used to develop the PlayCable adapter.

Secondly, although no record of the ROM can be found on the internet, it can be deduced that it has the same pinout as a standard 1K x 8 JEDEC ROM. Additionally, only the lower 5 bits of the ROM are connected to the rest of the circuit. Therefore, like the PlayCable's RAM, the ROM is being "double pumped", with each Master Component request retrieving 5-bits of data from two locations in the ROM, that are then concatenated together into a byte. This explains why the MCLK signal is required for the correct operation of the ROM, and why standard ROM dumpers cannot be used with the PlayCable. Without a 3.58MHz MCLK signal, the state machine logic will not be able to operate faster than the Master Component and retrieve data from two addresses in one time slice.

Finally, although this design makes use of a PROM or mask ROM, the board is also designed to support EPROMs. The pins needed to control the program state of a 2708 JEDEC EPROM are made available on two pads. Like the use of standard components, this suggests that the board design is very close to the prototype PlayCable hardware. Furthermore, there is some evidence that instances of Jerrold PlayCables fitted with EPROMs also exist. Figure 33 shows an image of a Jerrold PlayCable posted in this AtariAge thread:

<http://atariage.com/forums/topic/203583-intellivision-playcable/>

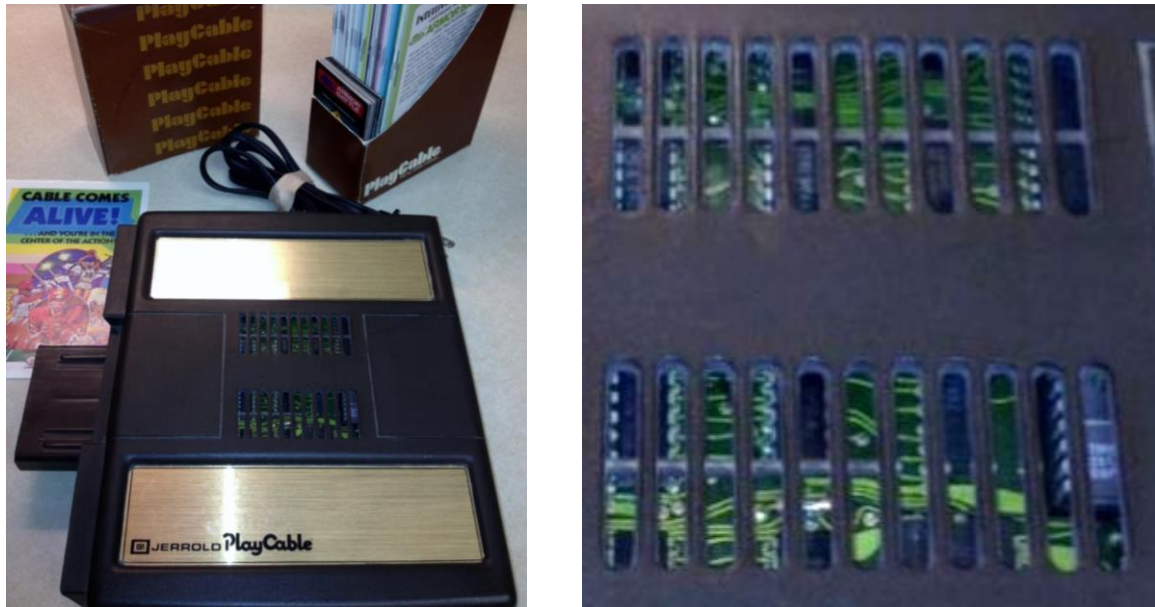


Figure 33 : AtariAge Jerrold PlayCable

In this image, the mesh that normally obscures the PlayCable's internals is not present, allowing the PCB to be clearly seen through the cooling vents. Once magnified, the layout visible through the vents is consistent with the upper board of intvsteve's machine. However, the grey ceramic ROM, visible lower right in Figure 33, appears to bear markings more commonly seen on an EPROM than a ROM. This chip also looks as though it might be mounted in a socket, rather than being soldered to the board.

The Jerrold PlayCable design had potentially serious ramifications for Mattel. An Intellivision peripheral was in the field that contained a serial interface, RAM and a small ROM, all based on standard, documented and easily obtained components. As shall be shown, the Jerrold PlayCable is a hacker's paradise. It would be possible to replace the PlayCable bootstrap ROM with other code, blown to a standard EPROM. Doing so would remove the need to reverse engineer the cable interface or communications protocols. It would then be possible to disconnect the UART from the demodulator and either directly interface the PlayCable to a computer's parallel port; or add a level shifter and change the UART clock to a more RS-232 friendly frequency, allowing connection to a serial port. Either way, with some technical knowledge it would be possible to connect a PlayCable to virtually any home computer and download software to it. Mattel would not be protected by the obscure nature of the PlayCable communications protocol, or the lack of EPROMs compatible with the CP-1610 bus.

Therefore, there is strong evidence that the Jerrold PlayCable pre-dates the General Instrument design and that migrating to the General Instrument ASIC implementation can be seen to serve two important purposes. Firstly, to significantly reduce the cost of PlayCable manufacture, and secondly, to raise barriers to hacking, by denying access to the bootstrap ROM and other internals of the circuit.

8.2 Jerrold Engineering Audio PlayCable

It has been discovered from conversations with Joe Jacobs and Dennis Clark that engineers working on PlayCable development hacked early Jerrold adapters. The engineers made use of the Jerrold head-end DataChannel broadcast cards and modified PlayCable adapters to gain unrestricted, free access to the PlayCable Intellivision library at home. The engineers found that it was possible to record the 14KHz digital signal used to encode Intellivision games onto a regular audio cassette. This was done by tapping the digital output of a PDP-11 broadcast card and passing it through an attenuation box to a tape recorder, as shown in Figure 34.

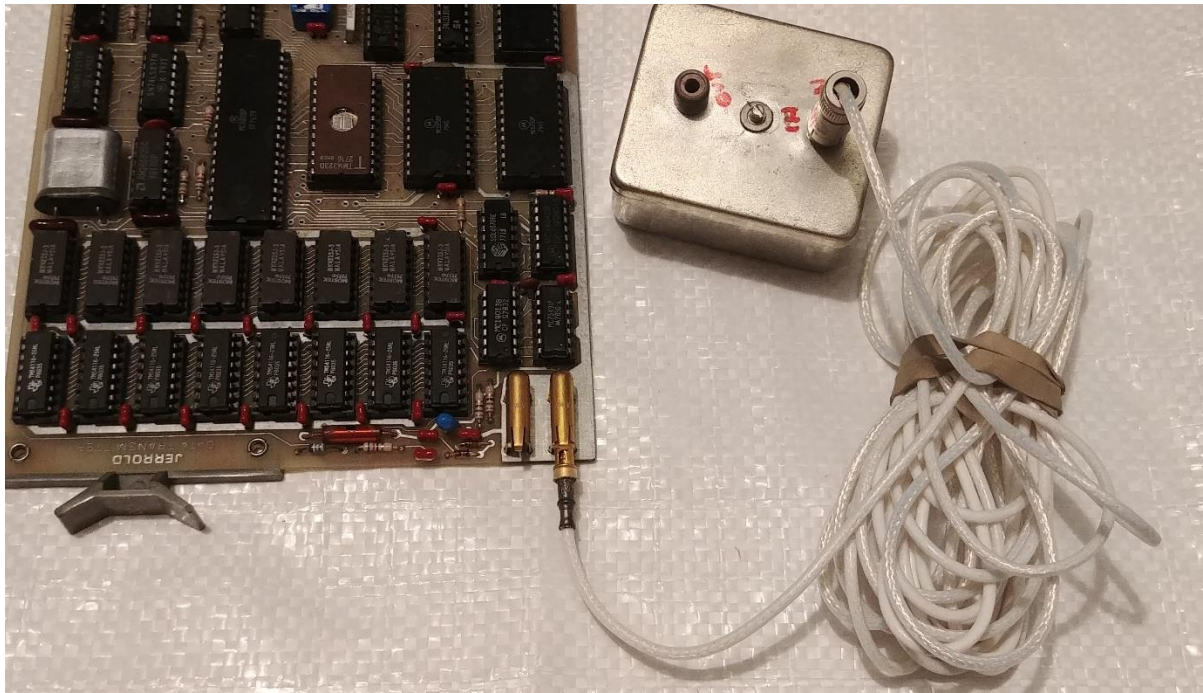


Figure 34 : DCX11A Broadcast Card and Audio Attenuator Box

Several streams, each containing a single game, were recorded one after another onto a single audio tape. This tape was then played into a hacked Jerrold PlayCable adapter at home.

It is believed that the changes to the PlayCable adapter bypassed the RF cable receiver and probably injected the audio signal at the equivalent of the output pin of the MC1351P FM receiver (pin 2). This is the point in the regular circuit that the 14KHz digital signal is first isolated in the normal processing chain. As noted in Section 6.6, the regular signal is essentially audio at this point.

It is also suspected that the hacked PlayCables may have run altered firmware that bypassed searching for the menu program and name checks normally performed when loading a game. Alternative firmware would have allowed the PlayCable to load the first program fed to it from tape, in a manner similar to executing LOAD "" on an 8-bit micro-computer.

Both of these points have yet to be confirmed through examination of a suitable unit.

8.3 Joe Jacobs / Dennis Clark Development System PlayCable

Famously, Joe Jacobs and Dennis Clark, two of the engineers at Jerold, took the hacking of PlayCable adapters to the next level, and in doing so created an Intellivision development environment. Joe still has one of these development PlayCables and has kindly provided images of it. Like the audio hack PlayCables, the development system created by Joe and Dennis was based around the Jerrold version of the PlayCable adapter. These units were changed to allow them to be directly connected to a PDP-11 host machine using an RS-232 serial interface. As can be seen in Figure 35 much of the PlayCable receiver board was stripped of semiconductor components leaving just the RAM, power supply and clock circuit of the original lower board intact. A large number of patch wires were also added to the digital board. As shall be shown, the primary reasons for these changes was to add two-way digital communication via RS-232, and to support the use of a debugger held in a larger capacity EPROM, rather than the standard firmware ROM.

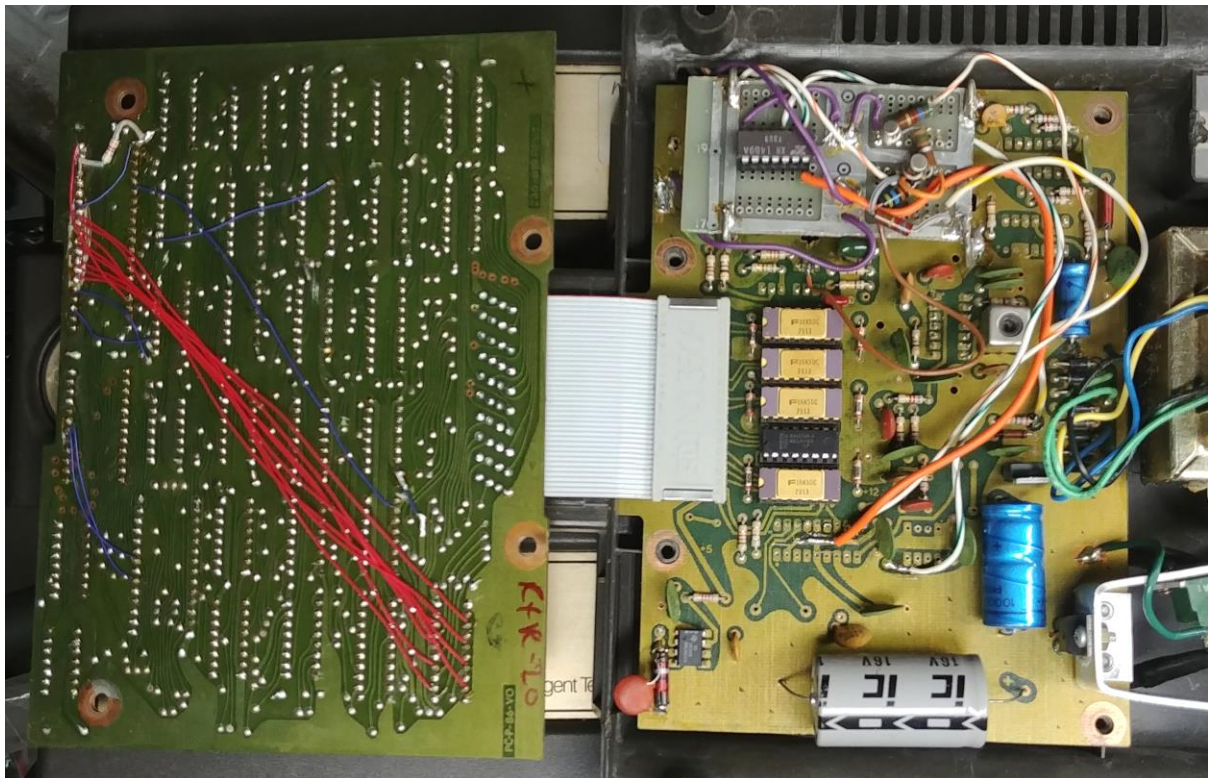


Figure 35 : Joe Jacobs / Dennis Clark Development PlayCable

Figure 36 shows a close up of the stripped receiver board. The daughterboard, located where the RF cable receiver is normally found, contains an XR1489A quad line receiver chip, a couple of transistors and passive components. The XR1489A is a common chip used to interface the incoming higher voltages of the RS-232 line to the TTL logic of the PlayCable. The transistors and passives on the daughter board do the reverse, boosting the outgoing logic voltages to RS-232 line levels.

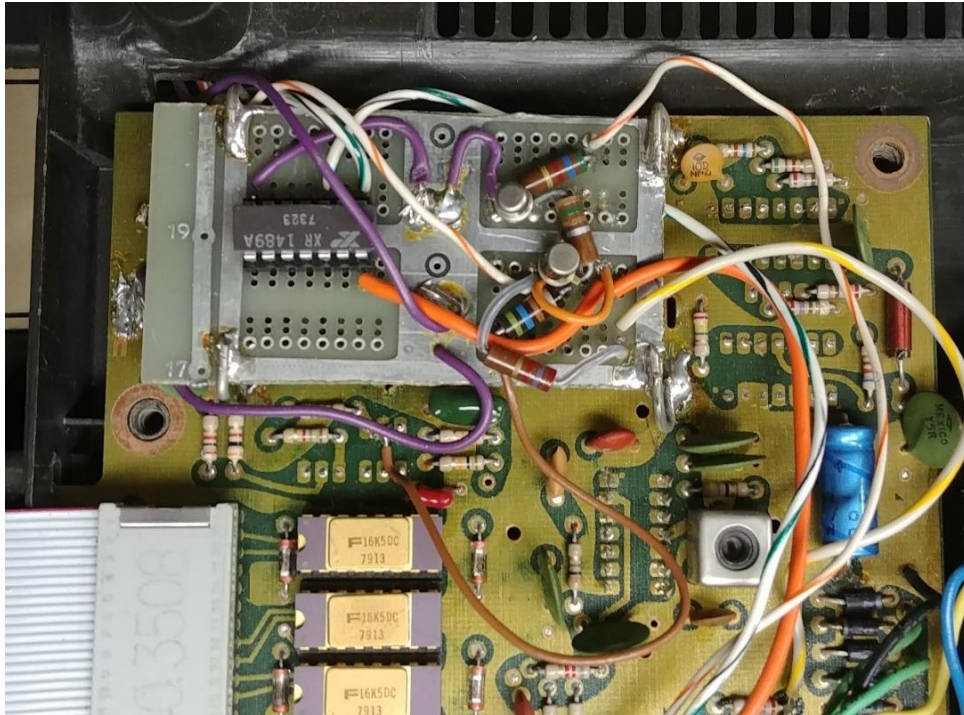


Figure 36 : Joe Jacobs / Dennis Clark Receiver Board

Figure 37 contains a speculative reverse engineering of the receiver (left) and transmitter (right) circuits employed by Joe and Dennis based on the images above. Whilst these are plausible, they may be incomplete / inaccurate as the reverse of the daughterboard is not visible, and the wiring of the transistors is partly obscured by their cans.

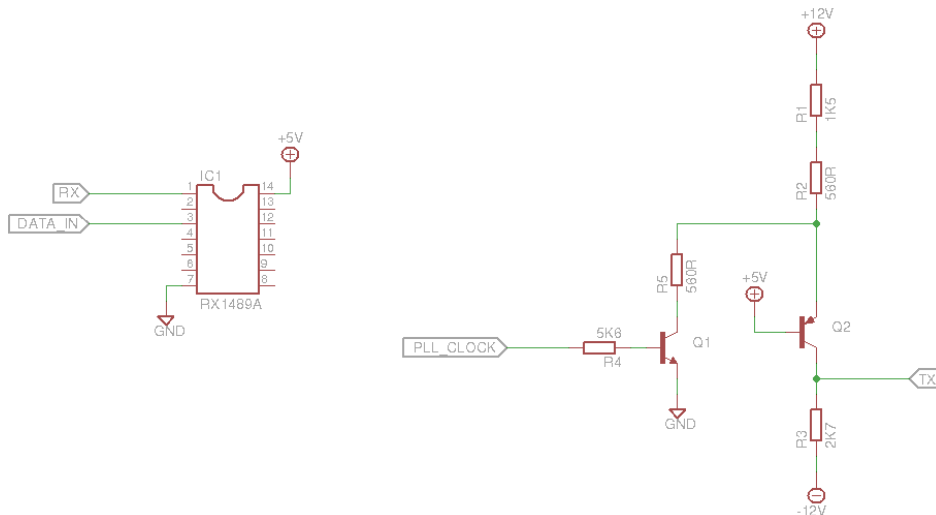


Figure 37 : Joe Jacobs / Dennis Clark Transmitter / Receiver Circuit

What is clear from the images is that the PlayCable development system only makes use of transmit, receive and ground wires, with no hardware flow control. Inbound data from the PDP-11 is passed to the DATA_IN signal of the digital subsystem and the PLL_CLOCK signal is repurposed to send data out to the RS-232 link through a simple inverting transistor amplifier.



Comparing the board changes with the AY-3-1015 pinout from a datasheet, the intention of the patches becomes clear. The red wires passing diagonally across the board link the DB1-DB8 signals that load an outgoing word to the PlayCable data bus. This, combined with connecting the DATA_STROBE line to a suitable write enable signal, allows the Master Component to load a word to the UART for transmission down the RS-232 link. The outgoing transmission line is taken from SERIAL_OUT on pin 25 and feeds the PLL_CLOCK line seen in Figure 37.

PIN CONFIGURATION 40 LEAD DUAL IN LINE

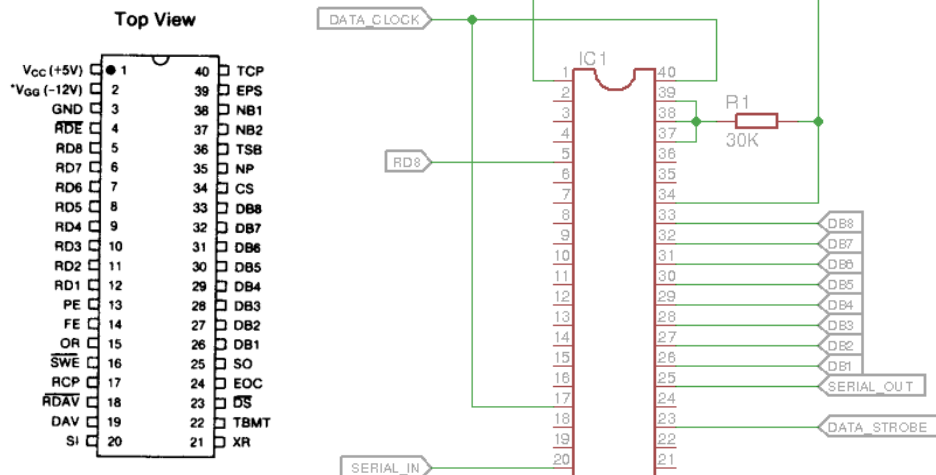


Figure 40 : Joe Jacobs / Dennis Clark AY-3-1015 Changes

Connecting pins 40 and 17 together means that transmission and reception occur at the same data rate as they then share a single clock. Tying pins 34, 37, 38 and 39 high configures the UART for 8-bit, even parity communication, this is in contrast with the PlayCable's standard mode of 7-bit, odd parity. Because all 8-bits are now used for incoming communication it is necessary to hook up the inbound DB8 line on pin 5 to the PlayCable data bus to make it available to the Master Component. The source of the serial input from the glue logic to pin 20 of the AY-3-1015 is also changed by a patch wire on the component side of the board (not shown). This is now routed directly from the DATA_IN pin via a 74LS367 buffer, bypassing the logic of the original Jerrold design that removed the Manchester encoding usually found in a PlayCable data stream.

The patches to the ROM / EPROM can be evaluated in a similar manner. Although not clear from the images, it can be inferred that, like the standard firmware ROM, the EPROM is a JEDEC compatible device. The two blue patch wires, visible at the top centre/left of Figure 39, link pins 18 and 19 of the EPROM to pins 22 and 20 of the MC3242A memory controller respectively. This suggests that the EPROM is a 2532 4K x 8-bit component and as such the firmware it contains could be as large as 2K decles in size. This is much larger than the 512 decles of the PlayCable firmware, although it would still fit into the existing window from \$4800 to \$4FFF, and therefore, would not need changes to the address decoding logic.

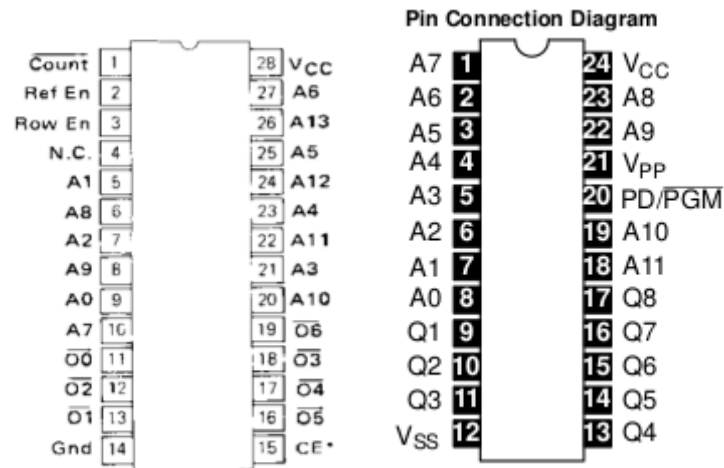


Figure 41 : MC3242A and 2532 EPROM Pinouts

Understanding the other small changes to the digital board glue logic require a more comprehensive reverse engineering of the baseline Jerrold adapter which has not been completed at this time.

8.4 Intellivision 2 Compatibility Modification

AtariAge user Lathe26 has provided images of the internals of a second General Instrument branded PlayCable in his possession. This PlayCable has some cosmetic differences, for example the replacement of the MK4315 RAM chips with the more common MC4116 and the packaging of the ASIC in a plastic DIP rather than a ceramic one, as shown in Figure 42.

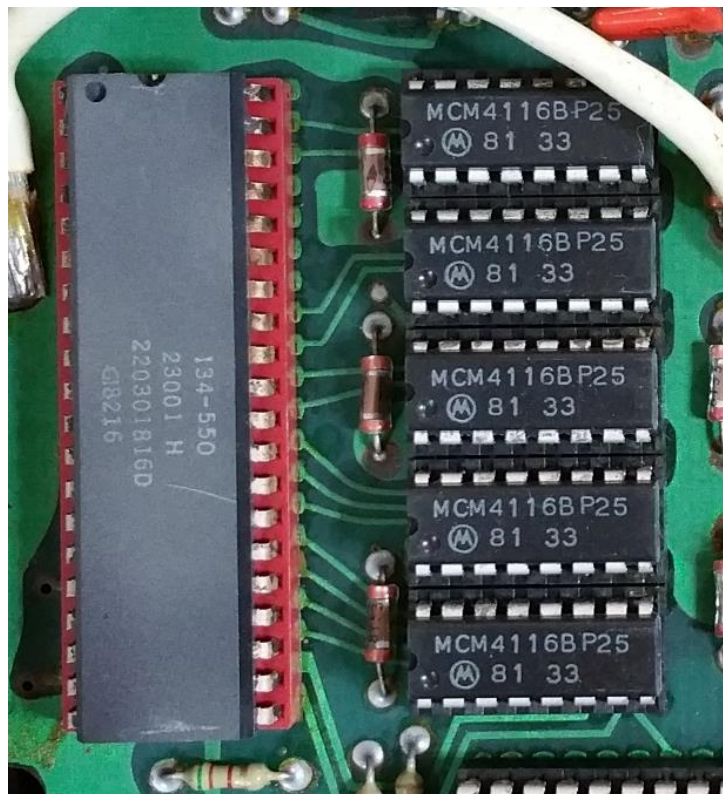


Figure 42 : PlayCable ASIC and RAM Variation

In addition, this model displays a couple of internal, post manufacturing alterations. The first is that two signals between the Master Component cartridge port and the ASIC have been crudely cut as shown in Figure 43.

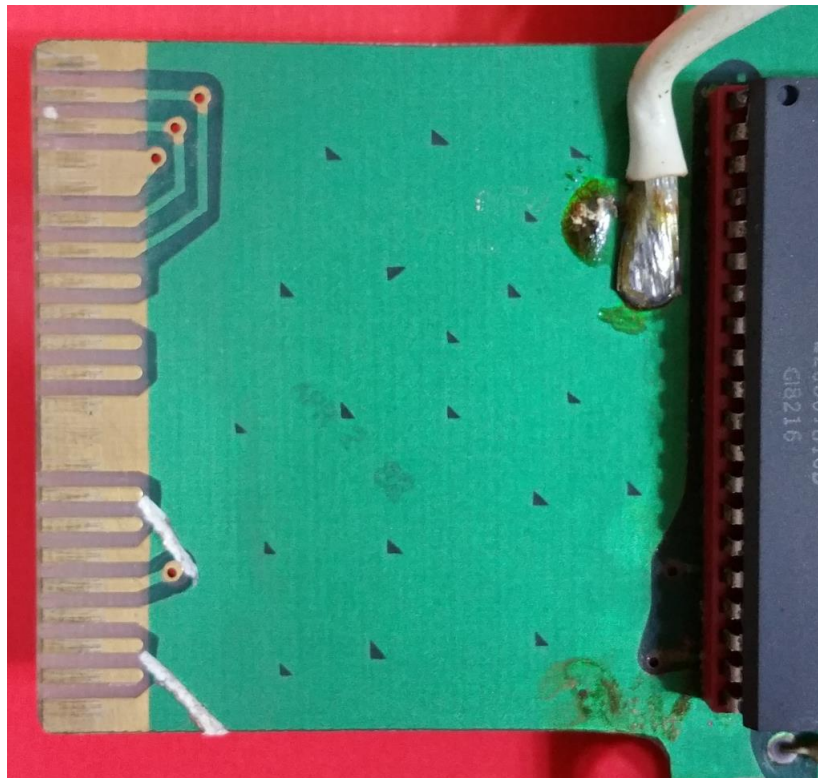


Figure 43 : PlayCable Cartridge Port Modification

The signals cut are pin 2, one of the three ground connections, which is used to feed external video from the Intellivision System Changer to the Intellivision II; and pin 12, the RESET input to the Intellivision Master Component.

The other change is an alteration to the clock circuit. This introduces a switch to change the value of C1 altering the impedance loading of the circuit on the Master Component, as shown in Figure 44. This switch's positions are labelled as 'I' and 'II' with the additional capacitor being active when the switch is in the position 'II'.



Figure 44 : PlayCable Clock Modification

Together these changes suggest that the PlayCable may have been found to be incompatible with some Intellivision II Master Components. However, talking to Paul Hilt, while Jerrold were concerned about, and did test the Intellivision II for compatibility, no issues were found. This finding has been confirmed by Lathe26, who indicates that Intellivision II units do not show any problems and the PlayCable is seen to work with the switch in both position I and position II. Therefore, it is clear that the changes were not part of the original manufacturing process, it is unlikely that they were subsequently made by Jerrold; and their ultimate purpose is unknown.

9 PlayCable Phoenix

The PlayCable Phoenix is a small board that has been constructed to resurrect the PlayCable and provide a sense of what it was like to use. The Phoenix moves the PlayCable ASIC onto a daughterboard that plugs into the vacated ASIC socket, as shown in Figure 45.

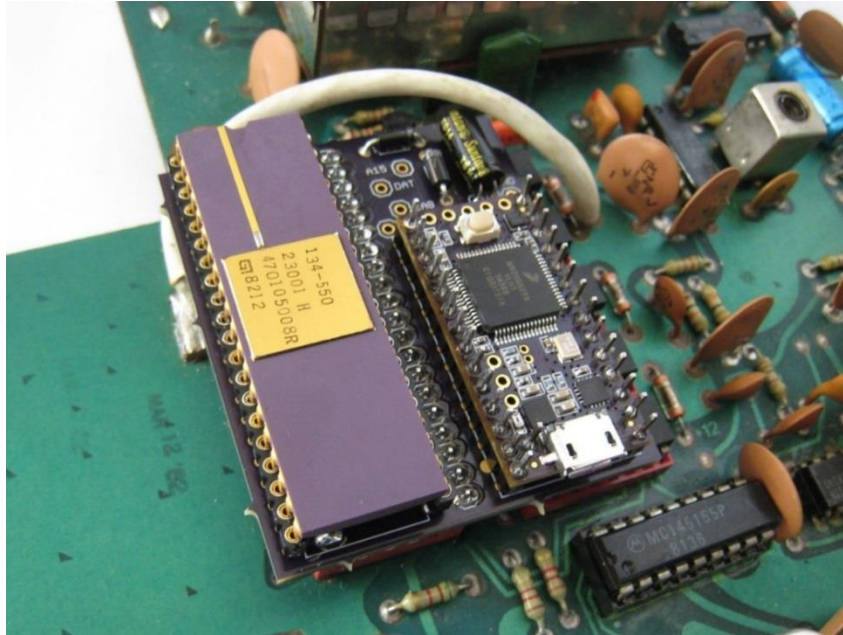


Figure 45 : PlayCable Phoenix

The modification is simple, reversable and non-destructive. The daughterboard disconnects the ASIC DATA_IN signal from the PlayCable demodulator and instead routes the output of a Teensy 3.2 microcontroller to it. The two ASIC outputs that feed tuning requests to the PLL tuner are duplicated and also fed to the microcontroller, as shown in Figure 46.

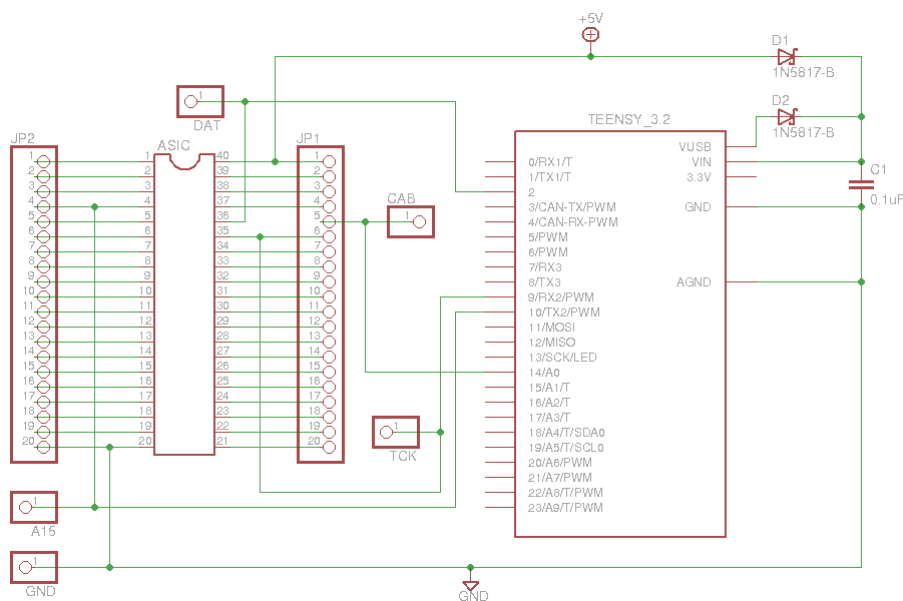


Figure 46 : PlayCable Phoenix Schematic

Firmware within the microcontroller listens to tuning requests generated by the ASIC and uses them to select which of 20 plus fake channel streams resident in its Flash ROM to transmit to the ASIC on the DATA_IN line. Typically, the channel streams comprise the replica menu program and all 20 games found in the March 1983 game roster. In addition, there are a number of unadvertised streams that demonstrate other features, including the correct operation of 8K games. The number and complexity of the programs made available by the PlayCable Phoenix are constrained by the 256K Flash memory capacity of the Teensy 3.2 and the 8K standard Intellivision memory map of the PlayCable ASIC.

A video of the operation of the PlayCable Phoenix can be seen here:

<http://atariage.com/forums/topic/285559-playcable-rising-from-the-ashes-demonstration/>