Inside DISTO's 2-Meg Kit
Tony Distefano

Inside the DISTO 2-Meg Kit

Introduction:

This brief will describe, in detail, how the DISTO 2-Meg Kit for the COCO 3 works. In order to fully comprehend the following brief, I will assume that you are already very familiar with CPU's, DRAM (Dynamic Random Access Memory), discrete logic gates, circuit diagrams and understand how the GIME chip of the COCO 3 works. It is also recommended that you have and understand the complete schematic diagram of the COCO 3.

Included is a full schematic diagram of both the DAT board and the Memory board as well as a full JEDEC listing of the PAL (Programmable Array Logic) chip. The discussion will start with a short overview on what is required to get the COCO 3 to recognize and use 2 Megabytes of memory. Then, a review of how the GIME works, followed by a theory of operation of DRAM refresh. Finally, there will be a 'chip by chip' circuit description and timings. Although I have spent a lot of time and effort into making this brief as accurate as possible, there is always a chance for errors.

Overview:

When I first thought about expanding the memory capabilities of the COCO 3, I knew it would require a lot of time trying to understand how the GIME treats memory. The 256K memory chips that the GIME uses has many working parameters. I wanted to increase it's (the COCO 3) memory capabilities to 2-Megs of memory. At the time, the most common memory upgrades were 1-Meg SIMMs. I decided to use two of them to make my upgrade. I also wanted my upgrade to work with as many types of SIMM possible, so I got as many kinds as I could find at the time. These were the 2-chip, 3-chip, 8-chip and 9-chip kinds. I had to make sure that my circuit worked with all of them. Whatever the case, there were a few electronic differences between the

chips on the SIMMs and the 256K memory chips currently used by the GIME. The first is the extra address lines. There are 2 more address lines on a 1-Meg chip than there are on a 256K chip. The second is that the refresh requirements are also more than the GIME can cope with. The final is not really a electronic difference as it is a logic difference. The GIME is not set up to handle the extra memory. The GIME uses a "bank-switching" method to allow the CPU to access more than 64K, but the bank switching capability of the GIME does not go beyond 512K. An extension to the existing circuits has to be added. That, along with a few more GIME quirks, is what's required to allow the COCO 3 to access and use 2-Megs of memory. The following is everything I know about the GIME and my 2-Meg memory upgrade kit.

The GIME (ACVC):

The heart of the COCO 3 is a Motorola MC68B09E CPU. This CPU can directly access 64K of memory via 16 address lines. The COCO 3 also has ROM, RAM, I/O ports and a color video display circuit. Tying all of these parts together is the GIME (Graphics/Interrupt/Memory/Enhancer a.k.a. “Graphics Interrupt Memory Enhancement” or “ACVC – Advanced Color Video Chip”). It is the job of the GIME to supply all timings and memory mappings for the COCO 3. The GIME also includes the DAT (Dynamic Allocation Table or Dynamic Address Translation) and video graphic display hardware. The CPU has two operating speeds: 1Mhz or 2Mhz (actually .89Mhz or 1.78Mhz). The video circuits always run at the same speed of 2Mhz. Wait a minute, how can the CPU run at 2Mhz and the video run at 2Mhz? Well, the GIME is always running at 4Mhz (actually 3.579545Mhz, or simply 3.58Mhz – this value being chosen because it happens to be the frequency of the Color Burst signal in a composite NTSC video signal). The GIME interleaves CPU and video at 1:1 when the CPU is running at 2Mhz. When the CPU is running at 1Mhz, the GIME skips every second CPU cycle and does nothing. It takes a rest. The GIME follows this sequence, CPU access ->then Video access ->then no access ->then Video access and then repeats the same operation. When the CPU is running at 2Mhz, the 'no access' is replaced with another CPU access. The CPU's clock system, E and Q clocks, change from 1Mhz to 2Mhz and back, but all memory access timings stay the same at 4Mhz or about 4 I/O cycles per micro-second.

The GIME generates all system timing and does so with the help of a 28.63636Mhz crystal in the NTSC version or a 28.475 oscillator in the PAL version (the frequency is shifted down slightly in the PAL version for compatibility with the PAL encoder circuit). The GIME uses this crystal and a multitude of counters and latches to generate video text and graphics, memory management, dynamic allocation, memory mapping, I/O mapping, ROM switching and many other functions. This chip is so complex that it would require hundreds of logic gates to duplicate. The COCO 3 service manual describes the GIME in great detail. I will not attempt to duplicate that information here, but only review the sections of it that pertain to the 2-Meg Kit.

The fact that the GIME is such a complex device, and that it has many signals that are not available to the outside world via chip pins is what made the 2-Meg Kit so hard to design. If certain signals were available, then upgrading the COCO 3 to 2-Megs of RAM would have been relatively simple. Instead, two boards with a bunch of chips and many hours of R & D was required.

DRAM Refresh:

A quick review of DRAM is necessary to understand refresh. First, this type of memory uses a multiplexed address line technique to save on the number of pins required to access great amounts of data. It uses two pins to strobe in two sets of address data. The 256K memory chips require 18 address lines while the 1M memory chips require 20 address lines. Dividing each address set into two gives 9 address lines for 256K chips and 10 address lines for 1M chips.

The first signal is the RAS (Row Address Strobe) line. It is used to strobe in the first half of the address lines. The second is the CAS (Column Address Strobe) line. It is used to strobe in the other half. To save space on the chip surface and therefore reduce production cost, the memory cells of DRAM are made up of tiny capacitors. When the capacitor is charged above a fixed internal value, it is considered logic 1 or HI. When the capacitor is below this level it is logic 0 or LO. The shortfall of this technique is that the voltage across the capacitor slowly decays and drops down to below the internal threshold all by itself. To avoid this, the capacitor must be topped up every so often.

This is 'Refresh'. Most DRAM chips have several methods of refresh available. The easiest refresh mode is a simple read cycle. The GIME uses this method of refresh. The COCO 3 with 512K of memory uses 256K DRAMs. These chips require a 256 cycle refresh which must be repeated every 4mS. The GIME accomplishes this by doing 10 video read cycles during every Hsync (horizontal sync) period. Hsync happens every 63µS (actually 62.5µS), therefore 256 cycles takes just under 2mS to complete. This is well under the required 4mS for this type of DRAM.

Chip by Chip:

When reading this section, have the schematic diagrams beside you for quick and easy reference. I refer to them continuously. The schematics are included within this document, and as loose pages for convenience. Whenever I mention more than one signal and/or pin number together, they are always respectively related.

The DISTO 2-Meg Kit is divided into two sections: the *DAT board* and the *Memory board*. Let's start with the DAT board description. J1 gives the DAT board access to all the pins of the CPU. This is required in order to have access to all of the address, data and control pins of the CPU. J2 is the junction between the DAT board and the Memory board. There are only 4 signals required between the two boards. Two of them (pin #2 and #4) are the last two bits (D6 & D7) of the DAT memory. Basically these two bits act like a 512K bank switcher. With two bits, you can control 4 banks and 4 banks of 512K each equals 2-Megs of memory. The third (pin #3) is the E clock, required by the Memory board for proper timing and the fourth (pin #1) is the CPU/*VID output generated by the Memory board. There are two users of memory: The CPU is one and the video circuit is the other. This signal tells the bank switcher circuit whether the current cycle is a CPU cycle or a video cycle. More on that signal later. U6 is a PAL chip and is used strictly for memory mapping. It otherwise would have taken several standard logic chips to achieve the same results. The PAL used is a generic *18CV8*. All the signals to and from this chip are labeled according to function. U5 is also used for memory mapping. It selects the higher order of address lines into the PAL. With that, the PAL is able to address single locations of memory without mirroring. There are only four outputs from the PAL. The

first, 01, is pin #19. It is a write only memory select from $FFA0 to $FFAF ($FFAx). The second, 02, is pin #18. It is a write only memory select for only $FF91. The third, 03, is pin #17. It is a write only memory select for only $FF9B. The fourth, 04, is pin #16. It is a read/write select for $FE00 to $FEFF ($FExx). The remainder of the pins are taken up by the various address and control lines needed to properly decode these signals. A full JEDEC listing of this PAL is listed at the end of this brief.

U1 through U4 form the DAT memory D6 and D7. These 74LS670s are dual access memory chips. Each can access 4 bits in 4 locations. Since the DAT board only needs 2 bits, only two are used. They have separate input/output as well as separate address select pins. These chips have two separate modes of operation. The DAT Read/Write mode and the Memory Write mode. A similar chip is the 74LS170, which is an "*open-collector*" version, and would require *pull-up* resistors if used.

In the Memory Write mode, the CPU can write to these chips as if they were, 'write only' memory mapped. They are memory mapped to the same locations as the GIME's DAT memory, from $FFA0 to $FFAF. The data lines are connected to the missing D6 and D7. The DAT diagram labels which are which. Since the required memory is 16 address locations, 4 chips were needed. WA and WB of all 4 chips are connected to A0 and A1 of the CPU. U7A is a 2-TO-4 decoder used to decode A2 and A3 into 1 of 4 chip selects (GW). The enable pin of U7A (G) gets it's memory map from U6. In the DAT Read/Write mode, these chips function in the same way as the DAT memory in the GIME but only to D6 and D7 which do not exist on the GIME. RA and RB of all 4 chips are connected to A13 and A14 of the CPU. These along with A15 act as the DAT decoder. A15 and the output of U9A form the input to U7B which is a 2-TO4 decoder. (The output of U9A duplicates the TASK register in the GIME. This duplication is necessary because, the output of the TASK register is needed to reflect which TASK group is active, and is not available outside the GIME. The TASK register is D0 of address $FF91 and is write only in this circuit.) This selects which of the 4 chips is needed. In any video text or graphics mode, the GIME can access a

maximum of 512K of memory. The GIME can place video anywhere within this 512K area. The problem with 2-Megs of memory is that the GIME does not know how to deal with the extra memory. Built onto the DAT board is a circuit that divides the 2-Megs of memory into four 512K banks. U9B and Ul0A are 2 write only, latched bits, D0 and D1, memory mapped to location $FF9B. These two bits control these 4 banks of video memory. On power/up or reset, these two bits are 00. This lets the GIME access the first 512K bank of memory during the video cycle. Writing a number from 0 to 3 into this area will select which bank of video memory the GIME will use. This allows the user to switch banks at any time, in order to access any part of the full 2-Megs of memory for video. There is, however, one limitation. The GIME cannot switch banks on its own. This means that the user must make sure that any video text or graphics screen does not cross a 512K boundary.

The last chip on the DAT board is U8. This chip (74F153) is a dual 4-TO-1 encoder. This chip takes in the two DAT bits (D6 & D7, which basically represent four 512K banks) and the two video bank bits and encodes them into one of four modes. The four modes are controlled by inputs A & B of U8. The B input comes from the Memory board. Circuits over on the Memory board generate a signal called CPU/*VID. When this signal is HI (logic 1), the GIME is generating a CPU memory cycle. When it is LO (logic 0), the GIME is generating a VIDEO memory cycle. This signal is needed to select between the complex data of the DAT memory and the simple two bit data of the video bank bits. The A input of U8 roots from the PAL. This signal is active when the CPU accesses a read or write in the area from $FE00 to $FEFF or simply $FExx. The value for x is 'don't care' or 'any number'. When active, LO (Logic 0), this means that the CPU is accessing this $FExx area. There is a special condition when the CPU access this area, it must always be the same area, no mater how the DAT memory is set. When there was only 512K in the COCO 3, this was no problem, the GIME would make sure that this was so. But, with the addition of three more banks of 512K (4 banks total), a special condition had to exist to make sure that whenever this area was accessed, it would be the same. Bank 0. This A input insures this. However, this condition has to be true only when the CPU wanted that area. The video circuit is exempt of this condition. That is the fourth mode. The following is a description of the 4 modes of U8.

U8 Modes:

Mode 0: A=0, B=0
This is a video cycle while the CPU is accessing $FExx. In reality, this mode can never happen, but rather than having the inputs floating, they are tied to the Video bank bits.

Mode 1: A=1, B=0
This is a video cycle while the CPU is not accessing $FExx. Now this is the real video mode. Not only is the CPU not accessing $FExx but it is not accessing anything. The outputs of U8 are the data from the Video bank bits. In this mode, the GIME is accessing video data from one of the four 512K banks set by the two Video bank bits.

Mode 2: A=0, B=1
This is a CPU cycle while the CPU is accessing $FExx. In this mode the two output bits are forced to LO. This selects bank 0. The data in the DAT memory is being ignored and bank 0 is being selected. The data inside the DAT memory of the GIME is also being ignored and forced to 0, therefore always accessing the same 256 bytes of real RAM. Circuits inside the GIME are taking care of this. All this circuit has to do is present bank 0 during this mode.

Mode 3: A=1, B=1
This is a CPU cycle while the CPU is accessing memory other than $FExx. This is the mode when the data in the DAT memory switches to one of the four 512K banks. With the DAT memory inside the GIME and the DAT memory held in this adapter, the CPU can access anywhere in the 2-Meg address space.

That is just about it for the DAT board. Now we move over to the Memory board. This board has three duties. The first duty is to host the memory SIMMs. U101 and U102 are standard SIMM sockets. They each hold one 1-Meg SIMM. Now, I have tried just about every kind of SIMM I could get hold of in this board. The 8-Chip, 9-Chip, 2-Chip and 3-Chip all work with this board. I have tried from 120ns to 80ns speeds and they also worked. So, just about anything that is called a ' 1-Meg SIMM' works.

There are three connectors on the Memory board. J1'101 and J1'102 connect to the COCO 3's regular memory expansion connector. JP103 is the connector that shuttles over to the DAT board and to the two required signals on the COCO 3's motherboard.

The second duty the memory board is to multiplex the two DAT bits into A9 (explained earlier). This is done with one half of U104. U104 is a dual 4-TO-1 encoder, functionally identical to U8 of the DAT board. On the Memory board schematic you can see that DAT6 is connected to both 2C0 and 2C2. Similarly, DAT7 is connected to both 2C1 and 2C3. This limits this half of the encoder to only a 2-TO-1 encoder. Only the A input will switch signals. The logic level of the B input will not change the output. (I did this strictly as a trick to economize chips. I used the other half of this chip to do another function using only the B input. Otherwise, two chips would be needed.) The A input is used as a RAS/CAS selector. With proper timing from U103, when A is LO, DAT6 will become the RAS address of A9. When A is Hi, DAT7 will become the CAS address of A9. In other-words, DAT6 and DAT7 become the multiplexed address A9 of the 1-Meg SIMMs. The other half of U104 is used to alter the CAS signal to force the SIMMs into an 'internal refresh cycle' mode. The mode used is the CAS-Before-RAS Refresh. In this half, CAS from the COCO 3's GIME is routed to 1C2 and 1C3. 1C0 and 1C1 are grounded. This configuration also limits this half of the encoder to a 2-TO-1 encoder. The A input of this chip will have no effect on the output. Only the B input will select ground (logic LO) or CAS to the output 1Y. I will discuss the source of the B input in detail a little later. When the B input is HI, the output, LOUT, (which is connected to the CAS input of the SIMMs) will follow the CAS signal. When the B input is LO, the output, COUT, will also be forced LO.

We now know how U104 is used to control CAS and multiplex A9. The last four chips are used to generate the proper timing to control these go signals. These four chips represent the lion's share of time and effort to design. It is the heart of the 2-Meg adapter. In order to control CAS and to multiplex A9 at the proper time, all timing sequences had to be in sync, somehow, to the GIME. It would have been easier to design the 2-Meg adapter if certain signals were available from within the GIME, but they are not, therefore they have to be synthesized.

Three timing signals are needed. The first signal needed is to know when the CPU is accessing memory and Video is accessing memory. The second is when to switch between the RAS and the CAS address. The third is knowing, when and how, the GIME is doing refresh. To find out how something works without having the benefit of original diagrams and theories is called 'reverse engineering'.

After a lot of time spent with a logic analyzer, I found a common signal. All signals had a constant relation with the E clock. With the knowledge that everything is in sync with the falling edge of the E clock, this was a good point to start. Since the 28Mhz clock is the shortest time in which anything can happen, any timing edge or window can be derived. U103 is a loadable synchronous up/down counter. The clock to U103 comes from a buffered (Ul06B) 28Mhz. U105A is used to load a preset value into the counter. The output Q of U105A drives the Load input of the counter. Because the PR (preset) is connected to 28Mhz, Q is continuously being preset to its HI state. The CLK input to U105A comes from an inverted E clock via U106A because this gate uses the rising edge. Therefore, the falling edge of the E clock (inverted to the proper edge for this gate) will transfer D (which is connected to GND or logic LO) to Q. This will cause Q to go LO, but not for long, because the PR is being driven by the 28 Mhz. Q will only stay LO until the next LO cycle of the 28Mhz which is only about 18 nano-seconds (18ns) later, which, by the way is just about the speed limit of this counter. In other words, the falling edge of the E clock resets the counting operation of U103.

Now, what we have is a 4-bit counter in sync with the falling edge of the E clock. This 4-bit counter gives 16 discrete positions and 32 edges in which to choose the signals needed. Nothing in the COCO 3 happens faster than these positions and edges. Knowing that, the next step is to find out where in these 16 positions our signals are placed. We need just two signals. The first is a signal that will present the RAS address and the CAS address in proper timing and sync with the RAS and CAS of the GIME. (This signal is not known because it is not available outside the GIME.) This was done by looking at the 4 outputs of U103 and the RAS/CAS signals on a high speed oscilloscope. By setting up the scope to see the time between RAS going LO and CAS going LO on one channel and looking at each output one at a time on another, I was able to see which output fell exactly between them.

QB of U103 turns out to be the perfect signal. It allowed the RAS/CAS address to be stable and switch in the precise time that the GIME uses to control the other 9 address lines. More reverse engineering! The other signal needed in my circuit is the CPUVID selector. This signal has two uses. The first is to allow CPU access to DRAM during its Read/Write cycle and allows the VIDEO circuit access to DRAM during its Read cycle. With more scope watching of the E clock and the CAS signals, QD of U103 has exactly that needed output. It is HI during any CPU access and LO during any VIDEO access. This output is fed back to the DAT board to select the proper data via the B input of U8.

The second use of CPUVID is to trigger a refresh cycle for DRAM. To properly refresh DRAM is the third duty of the Memory board. As previously explained, the only option available to refresh DRAM without using many chips as counters and selectors is the CAS before RAS refresh. To use the internal DRAM refresh counters in this mode, all this circuit has to do is to force the RAS/CAS lines into a CAS before RAS refresh cycle.

Given that the GIME uses video cycles during the Hsync period to do its refresh (many hours of reverse engineering), this circuit uses the same period. Given that during the Hsync period, video is used to do refresh, this circuit uses these video cycles and converts them into CAS before RAS refresh cycles. The easiest way to generate this type of cycle is to extend the time normally given to a 'CPU only' cycle and create a 'double long' cycle. By creating this 'double long' 'CPU only' cycle, the video cycle is lost. No big loss there, they were used to do refresh anyway. Instead of the GIME doing a 'read only' refresh cycle, a CAS before RAS refresh cycle is generated. All this circuit does is to switch from a Read cycle refresh (using the GIME's internal counters) to a CAS before RAS refresh (using the DRAM's internal counters). This method allows the GIME to use the longer '512 cycle refresh' requirements of 1-Meg DRAM without having to extend the GIME's internal counters. After more scope gazing, I found that the rising edge of CPUVID was a good place to start. It is unaffected by the dual speed mode.

This edge is too early for CPU CAS, but the falling edge is just a little too tight for comfort. So, a short delay is needed. The rising edge of CPUVID drives the B input of U107A. On this edge, Q goes high until about the middle of the CPU CAS. The output of Q goes into the A input of U107B. The falling edge of Q (U107A) causes *Q of U107B to go LO. The length of *Q is adjusted so that it stays LO until the middle of the VID CAS.

To sum up, *Q, pin #12 of U107B, goes LO from the middle of the CPU CAS to the middle of the VID CAS. This output drives the B input of U104. Now, this half of U104 is configured as a two to one multiplexer, because of the way it is wired. When B is HI, CAS (pin #3 & #4) is output to COUT. When B is LO, GND is output therefore forcing CAS to stay LO during the time between the CPU CAS and the VID CAS. This action forces the DRAM chips to do an internal refresh cycle.

Now this type of cycle cannot go on forever, there would be no video otherwise. There is a way to control this. Since the GIME does refresh only during Hsync, I gated U107A to output a pulse only during Hsync. The A input of U107A is from Hsync of the GIME. When A is HI, no pulses are generated, therefore no double long CAS, therefore normal operation of CAS. When A is LO, during Hsync, pulses are generated, therefore double long (refresh) cycles are in effect.

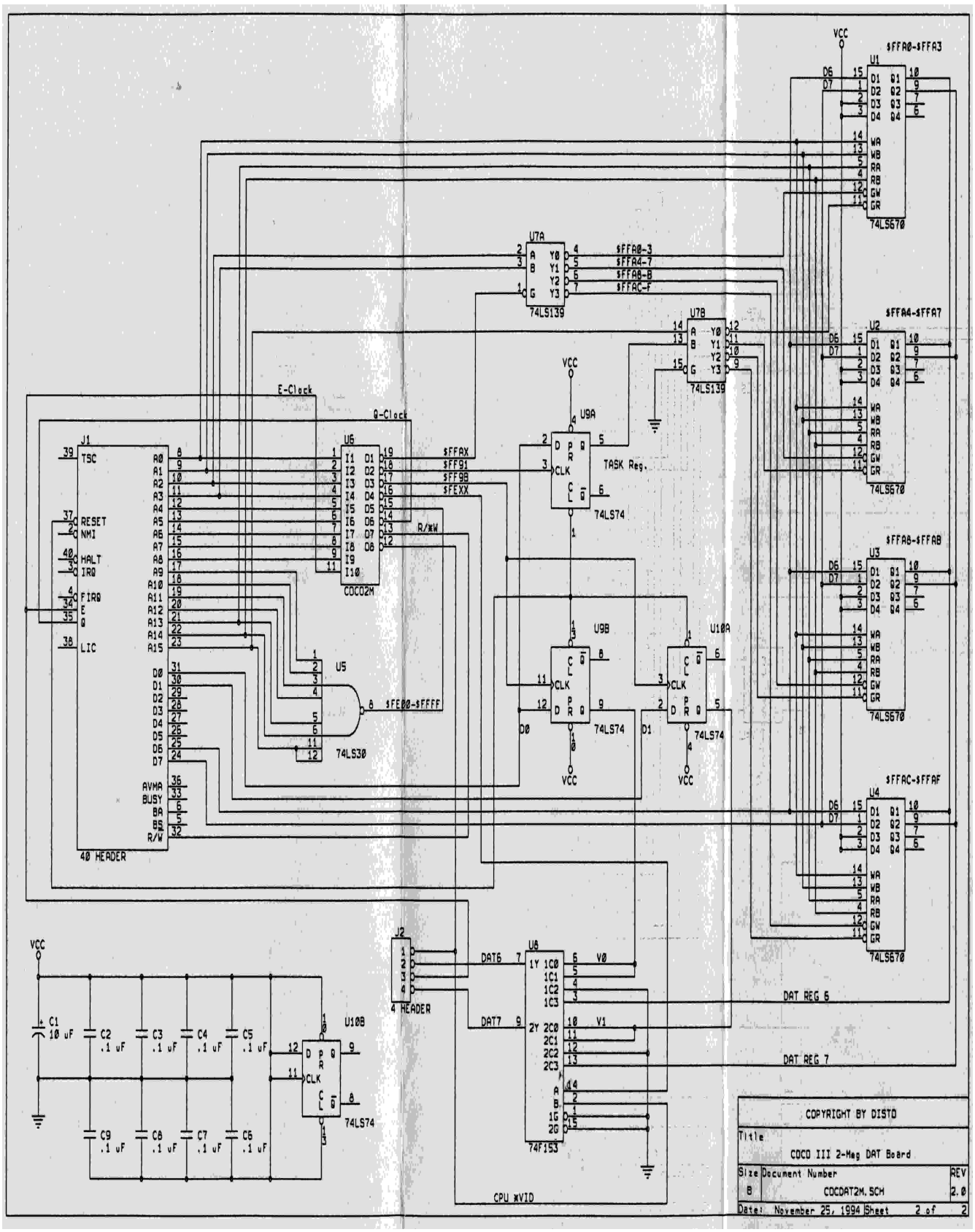
Conclusion:

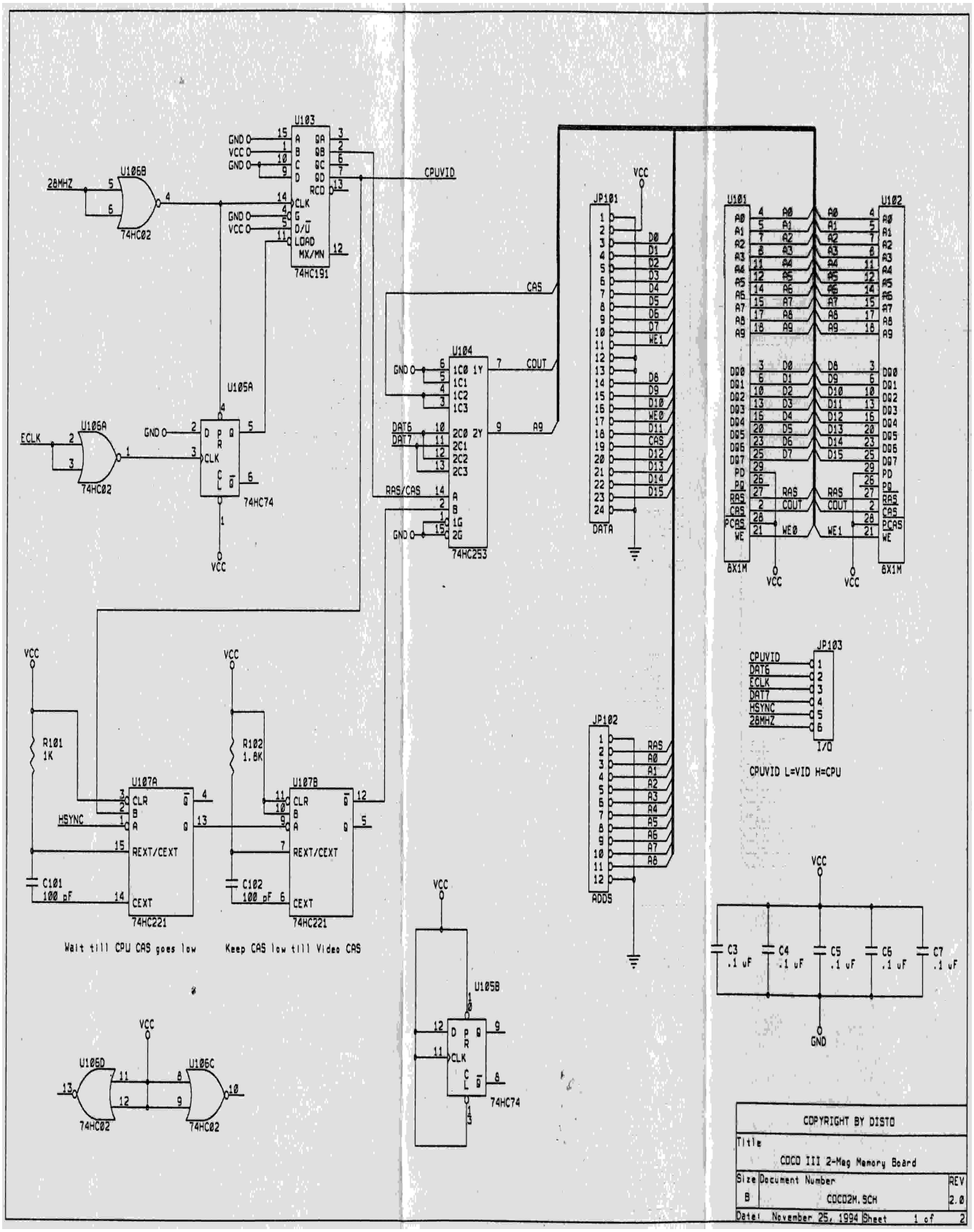
Well, that's about it. When you put all of these circuits together in the COCO 3, it can access 2 Megabytes of memory.

JEDEC PAL Listing

```
Name COCO2MEG;
Partno DIS00020;
Date 02/10/93;
Revision 02;
Designer Tony DiStefano;
Company DISTO;
Assembly DAT Board;
Location U6;
/************************************************/
/* */
/* COCO2MEG memory map decode logic */
/* */
/************************************************/
/* Allowable Target Device Types: PAL16L8 EP320 */
/************************************************/
/** Inputs **/
Pin [1..9] = [A0..8] ; /* CPU Address Lines */
Pin 11 = E ; /* CPU E CLOCK */
Pin 13 = !WR ; /* CPU Write when LO */
Pin 14 = Q ; /* CPU 0 CLOCK */
Pin 15 = !AX ; /* 1111,111X,XXXX,XXXX = LO */
Pin 12 = CPU ; /* CPU = HI VIDEO = LO */
/** Outputs **/
Pin 19 = !FFAX ; /* Write Data to REGISTERS */
Pin 18 = !FF91 ; /* Task Register Bit */
Pin 17 = !FF9B ; /* Video Bank Bit */
Pin 16 = !FEXXM ; /* Disable REG when $FEXX */
/** Declarations and Intermediate Variable Definitions **/
/** Logic Equations **/
FFAX = Q & E & WR & AX & A8 & A7 & !A6 & A5 & !A4;
FF91 = Q & E & WR & AX & A8 & A7 & !A6 & !A5 & A4
& !A3 & !A2 & !A1 & A0;
FF9B = Q & E & WR & AX & A8 & A7 & !A6 & !A5 & A4
& A3 & !A2 & A1 & A0;
FEXXM = AX & !A8 & CPU;
FFAXW = E & AX & A8 & A7 & !A6 & A5 & !A4;
```

Note: The above will not properly compile in WinCUPL for a 16V8 GAL (g16v8;). It is intended for an 18CV8. Changes to the schematic will allow use of a (g20v10;) or (g22v10;)


VCC
$FFA0-$FFA3
U1
74LS670
U7A
74LS139
$FFA0-3
$FFA4-7
$FFA8-B
$FFAC-F
U7B
$FFA4-$FFA7
U2
E-Clock
Q-Clock
J1
40 HEADER
U6
COCO2M
$FFAX
$FF91
$FF9B
$FEXX
R/W
U9A
TASK Reg.
74LS74
U5
74LS30
$FE00-$FFFF
U9B
U10A
D0
D1
$FFA8-$FFAB
U3
$FFAC-$FFAF
U4
J2
4 HEADER
U8
74F153
DAT6
DAT7
V0
V1
DAT REG 6
DAT REG 7
CPU xVID
U10B
C1 10 uF
C2 .1 uF
C3 .1 uF
C4 .1 uF
C5 .1 uF
C6 .1 uF
C7 .1 uF
C8 .1 uF
C9 .1 uF
COPYRIGHT BY DISTO
Title
COCO III 2-Meg DAT Board
Size B
Document Number
COCDAT2M.SCH
REV 2.0
Date: November 25, 1994
Sheet 2 of 2


U103
74HC191
U106B
74HC02
28MHZ
CPUVID
U104
74HC253
CAS
CDUT
A9
RAS/CAS
DAT6
DAT7
U105A
U106A
74HC74
ECLK
JP101
DATA
U101
U102
8X1M
JP102
ADDS
JP103
CPUVID
DAT6
ECLK
DAT7
HSYNC
28MHZ
I/O
CPUVID L=VID H=CPU
R101
1K
R102
1.8K
U107A
U107B
74HC221
HSYNC
C101
100 pF
C102
100 pF
Wait till CPU CAS goes low
Keep CAS low till Video CAS
U105B
U106D
U106C
C3
.1 uF
C4
.1 uF
C5
.1 uF
C6
.1 uF
C7
.1 uF
GND
VCC
COPYRIGHT BY DISTO
Title
COCO III 2-Meg Memory Board
Size
B
Document Number
COCO2M.SCH
REV
2.0
Date: November 25, 1994
Sheet 1 of 2

DISTO

2-MEG MEMORY

UPGRADE KIT

Congratulations on the purchase of your new DISTO 2 Meg Memory Upgrade Kit. The quality materials and workmanship used in this product insure years of trouble free use. This product will only work with a COCO III computer. Please read this manual completely before opening your computer. It is very important that you understand what you are doing before you start. Good soldering experience is needed. **WARNING!** Opening your computer may void your warranty. Not knowing what you are doing may void your computer. After reading all the instructions, if you feel that you cannot do the installation, contact someone that can. We at DISTO are not responsible for any damage to your computer. We have made every effort possible that this manual is accurate.

FEATURES.

- Transparent to all software that supports extra memory.
- Needs no other memory upgrades or cards.
- Fits completely inside the COCO III case.
- Easy installation - installs in an evening.
- Includes all necessary parts except memory chips.
- Drivers for OS9 included.

Theory of Operations.

The COCO III uses the MC68B09E CPU. This is an 8-bit CPU that can address 64K of memory. Before you bought this upgrade kit, the COCO III could access a maximum of 512K memory using a page switching method via the GIME chip. Now, with this kit, the page switching capabilities of the GIME chip has been extended to 2 Mega bytes of memory. This is done using two small internal boards. The first is the "Memory" board which is a special board that use two 1 meg SIMMs and plugs into the memory expansion socket of your COCO. The second is the "DAT" (Dynamic Allocation Translation) board which is a page switching circuitry that adds the extra memory to the capability of the GIME. This method of page switching becomes transparent to software capable of using the extra memory. Note: before the software can make use of this extra memory, it must know of its existence. Check with the software's manual for its ability to use the complete 2 Meg.

Installation of the DAT Socket.

Before you start this installation, make sure you have the following; SOLDERING EXPERIENCE, a fine-tip soldering iron, small wire cutters, solder, thin solid wire, star screw driver and a continuity checker. Make sure that you have a large, clean and static free working area. All references to, position, are taken when the computer is in the normal typing position.

1 - Unplug everything to your COCO and remove the top cover. Remove the keyboard by carefully unplugging the connector. Use slow even pressure on both sides.

2 - Locate R22 just above the keyboard adapter. Use a 1" piece of this wire and solder one end to one side of this resistor. Solder the other end of the wire to the other end of the resistor. This shorts out R22.

3 - Locate the CPU labelled IC1. It is next to the cartridge connector. If your CPU is not in a socket go to step #4. Carefully remove the CPU chip from its socket and insert it into the socket of the DAT board. Remember that pin #1 of the CPU chip must go into pin #1 of the socket. Pin #1 of the DAT board is just left of the J1 label. Insert the supplied 40 pin socket into the now vacant IC1 socket making sure that pin #1 of the socket goes into pin #1 of the IC1 socket. This serves two purposes: it protects the original socket from getting damaged and raises the socket so to clear the surrounding components. Now, go to step #9.

4 - Examine the supplied 40 pin socket included in this kit. Notice the 20 pins on each side. They have the same spacing as the CPU. Carefully place the socket on top of the CPU. Make sure that pin #1 of the socket aligns with pin #1 of the CPU. If necessary, slightly pry open one side of the socket so that the pins of the socket lean against the pins of the CPU. Each pin of the socket must press on each CPU pin. The socket must rest flat on top of the CPU. Make sure that all pins are positioned properly and that there are no leftover pins on either end.

5 - Carefully solder each socket pin to the corresponding CPU pin one at a time. Make sure that you do not use too much solder. Make sure that the solder flows from the socket pin to the CPU pin and that it does not short out to any adjacent pin. Solder all 40 pins being careful not to melt the socket or any other part with the hot soldering tip.

Testing the Socket Installation.

Pin #1 of the socket is now pin #1 of the CPU. Pin #2 of the socket is pin #2 of the CPU and so on. You now need to use your continuity checker to check your work. There are many brands of continuity checkers. You must know how to use yours. What you are trying to determine in the following instructions are two things. First, no shorts between pins, and second, continuity between the Adapter and the CPU.

6 - Make a visual test to see that each pin of the socket is properly soldered to the CPU. Check that there is no short between pin #1 and pin #2 of the Adapter. Some continuity checkers may show some resistance between these two pins, this is normal, what you are checking for are dead shorts or close to 0 ohms. If no shorts appear, check pin #2 and pin #3 and if no shorts appear, check pin #3 and pin #4 and so on until all pins are done. If there happens to be a short between pins, check your soldering where the adapter pins meet the CPU pins. Remove excess solder.

7 - Starting with pin #1, put one probe of your checker to the top part of the socket. With the other probe, touch pin #1 of the CPU. This is tricky because you must make sure that you do not touch the socket pin. Use a safety pin for a probe if your probe is too thick. This test must show that there is continuity. Check all pins in this way. If you come across a pin that does
show continuity, reheat it with your iron.

8 - The final test for this section is to plug in the computer and turn it on. If you do not get your regular screen, turn it off right away and go back to step #6. Check your computer with several pieces of software and all your peripherals plugged in. When you are satisfied that all is working well, continue the installation.

9 - Insert the long pins of the DAT board into the socket making sure that pin #1 of the DAT board enters pin #1 of the socket. Pin #1 of the DAT board is just left of the J1 label. Rock the board slowly and lightly, back and forth until all the pins are secure in the socket. Never press too hard on the DAT board to get it in! This

may damage the socket, the COCO III PC board and/or CPU.

10 - Soldered to the DAT board is a multi-conductor cable. Four of these wires are soldered to the DAT board and two are loose. These two must be soldered to different points on the COCO III board. The first point is the left-hand leg of the capacitor C60. The side that connects to the crystal X1. Solder the end wire to this point. The other point is pin #40 of IC5. Solder the second to end wire to this point. For now, place the 6 pin connector in such a way as to not touch anything.

11 - Plug in the computer and turn it on. If you do not get your regular screen, turn it off right away and check over all your work.

Installing the Memory Board.

You are now well on your way to a 2 Meg COCO III. The next few steps involves installing and testing the Memory board. Be patient, it's almost done.

12 - If your computer has a 512K memory board, remove it and go to step #13. Remove the memory chips in sockets labelled IC16, IC17, IC18 and IC19. Using a pair of cutters, remove capacitors C65 and C66.

13 - If your 2 Meg memory board came with 0K, you must acquire and install two 1 Meg x 8-Bit DRAM SIMMs. This must be the CMOS 2 chip type of 100ns or less. The CMOS 3 chip (1 Meg x 9-Bit DRAM) SIMM will also work. Make sure that pin #1 of the SIMM goes into pin #1 of the SIMM sockets U101 and U102 of the Memory board.

14 - Now, insert the Memory board into the white sockets of the COCO III labelled CN4, CN5 and CN6. There is only one orientation that will fit all three connectors. Make sure that all pins align and that there are no socket holes without pins. Do not insert the Memory board too deep into the connectors. This may cause shorts.

15 - Insert the 6 pin connector from the DAT board into the 6 pin header on the Memory board led JP103. Again, make sure that pin #1 of the connector goes into pin #1 of the header. Pin #1 of this connector is the end that goes to the DAT board, opposite to the loose ends that you have soldered in step #10.

16 - Connect the monitor and keyboard and turn the computer on. If a normal screen does not come on, turn the computer off right away and check your work from the beginning. LOAD and RUN the MEMTEST program. After a few minutes, you should see that you now have 2 meg of memory.

Technical Information.

The COCO III uses a chip called the GIME. The following discussion presumes that you have a good knowledge of the GIME chip, its functions and its memory map. As you know, the GIME has 16 DAT (Dynamic Address Translation) registers. Any of these 16 registers can contain a number from 0 to 63. This number, 0 to 63, is known as a Ram Block. Each of these blocks consists of 8K of Ram giving you a total of 512K of memory. In a HEX value, 0 to 63 is represented by 6 bits.

What the DISTO Memory Upgrade does, is add 2 more bits to the Ram Block. The value that can be entered into the DAT Registers is now 0 to 255. The locations are exactly the same as the GIME DAT

registers, from $FFA0 to $FFAF. But, unlike the GIME's DAT registers, you cannot read back the extra bits. This is no problem, since OS9 Level II and most other RS DOS programs do not need to read them. It is up to the software using the DAT registers to keep track of what is in each register.

Since there is no way for the GIME chip to know that the extra memory exists, it can not put any video in that area. It also can not cross any 512K boundary with video. But, there is a circuit that allows you to move the video completely into other 512K banks. This is done by adding a register into an unused memory area of the GIME chip. This location is at $FF9B. On power-up or reset, bit #0 and #1 of this location is set to 0. Writing a 1 to that location will force the video to come from the second 512K bank. Writing a 2 to that location will force the video to come from the third 512K bank and a 3 will be the fourth bank. Again, this is a write only location. It is up to the software to keep track of which bank the video is in.

If the Screen Start Address Register is set close enough to the end of a 512K boundary, the video will scan the beginning of the same 512K boundary and not cross over into the next 512K block. To avoid this, make sure that the start of video is in the proper location.

OS9 Operation

Complete OS9 drivers are included with this. When properly installed, the OS9 operating system will recognize a full 2 Meg of memory. There are no restrictions on how to use this memory. All programs should be able to make use of full memory. Complete instructions on how to install these drivers are included in a README file on the OS9 portion of the included disk. There may be more than one README file; if so, read all of them before starting. The instructions will require that you to make a new Boot disk. When making a new Boot disk, remember to start with only original Tandy modules. Then, when you get the 2 Meg working properly, you can start adding and modifying modules to customize your system to your needs. All OS9 drivers and patches are written and copyrighted by Kevin Darling unless other wise noted.

Credits:

The DISTO 2 Meg Upgrade Kit and all its documentation are conceived and designed by Tony DiStefano. The DISTO 2 Meg Upgrade Kit is manufactured and distributed by,

DISTO
1710 Depatie
St. Laurent, Quebec
Canada H4L 4A8

Further Reference: DISTO's 1-Meg CoCo 3 Upgrade

DISTO
SUPER PRODUCTS

1 MEG Upgrade Kit.

For the 512K COCO 3.

Manufactured and distributed by;

CRC Inc.
11 Boul. des Laurentides
Laval, Quebec
Canada H7G 2S3
1 (514) 967-0195

DISTO 1-MEG MEMORY UPGRADE KIT

Congratulations on the purchase of your new DISTO 1-Meg Memory Upgrade Kit. The quality materials and workmanship used in this product insures years of trouble free use. This product will only work with a 512K COCO 3 computer. Please read this manual completely before opening your computer. It is very important that you understand what you are doing before you start. Good soldering experience is needed. **WARNING!** Opening your computer may void your warranty. Not knowing what you are doing may void your computer. After reading all the instructions, if you feel that you cannot do the installation, contact someone that can. We at DISTO & CRC are not responsible for any damage to your computer. We have made every effort possible that this manual is accurate.

FEATURES.

* Transparent to all software that supports extra memory.
* Can use any other 512K memory card.
* Fits completely inside the COCO case.
* Easy installation - installs in 1 evening.
* Includes all necessary parts.
* Drivers for OS9 included.

Theory of Operations.

The COCO 3 uses the MC68B09E CPU. This is an 8-bit CPU that can address 64K of memory. Before you bought this upgrade kit, the COCO 3 could access a maximum of 512K memory using a page switching method via the GIME chip. Now, with this kit, the page switching capabilities of the GIME chip has been extended to 1 Mega Byte of memory. This is done using two small internal boards. The first is 512K of memory on a special board that allows you to install your already existing 512K board on top for a total memory of 1 Meg. Also on this board are; a power supply, so that the COCO's power supply does not overheat and a switching circuit to access the memory on the top board. The second board is the page switching circuitry that adds the extra memory to the capability of the GIME. This method of page switching becomes transparent to software capable of using the extra memory. Note: before the software can make use of this extra memory, it must know of its existence. Check with the software's manual for its ability to use the complete 1 Meg.

Installation of the CPU Adapter.

Before you start this installation, make sure you have the following; SOLDERING EXPERIENCE, a fine-tip soldering iron, small wire cutters, solder, thin solid wire, star screw driver and a continuity checker. Make sure that you have a large, clean and static free working area.

1 - Unplug everything to your COCO and remove the top cover. Locate the CPU labelled IC1. It is next to the cartridge connector.

2 - Examine the 40 pin adapter included in this kit. Notice the

20 pins on each side. They have the same spacing as the CPU. Carefully place the long and wide end of the adapter on top of the CPU. Pry open one side of the adapter so that the pins of the adapter are against the pins of the CPU. Each pin of the adapter must sit in the each CPU pin. The points of each pin on the adapter must rest on the main board. Make sure that all pins are positioned properly and that there are no leftover pins on either end.

3 - Solder each adapter pin to the corresponding CPU pin one at a time. Make sure that you do not use too much solder. Make sure that the solder flows from the adapter pin to the CPU pin and that it does not short out to any adjacent pin. Solder all 40 pins.

4 - Locate R22 just above the keyboard adapter. Use a 1" piece of this wire and solder one end to one side of this resistor. Solder the other end of the wire to the other end of the resistor. This shorts out R22.

Testing the Adapter Installation.

This finishes the soldering part of the installation. But, before you go on, you must make a few tests. Note: Pin #1 of the Adapter is pin #1 of the CPU. Pin #2 of the Adapter is pin #2 of the CPU and so on. You now need to use your continuity checker. There are many kinds of continuity checkers. You must know how to use yours. What you are trying to determine in the following instructions are two things. First, no shorts between pins and second continuity between the Adapter and the CPU.

5 - Check that there is no short between pin #1 and pin #2 of the Adapter. Some continuity checkers may show some

resistance between these two pins, this is normal, what you are checking for are dead shorts or close to 0 ohms. If no shorts appear, check pin #2 and pin #3 and if no shorts appear, check pin #3 and pin #4 and so on until all pins are done. If there happens to be a short between pins, check your soldering where the adapter pins meet the CPU pins. Remove excess solder.

6 - Using the same procedure as above check for shorts between pin #1 and the pin directly across, pin #40. Then check pin #2 and the pin directly across it, pin #39. Before going on to pin #3, check the pin before the pin across (pin #40) and the pin after (pin #38) with pin #2. Continue in the same manor for all the pins. This will make sure that there are no cross shorts. If there are shorts, check again your soldering around the pins in question, it is the only reason for shorts.

7 - For this test, it is necessary to remove the COCO's Printed Circuit Board (PCB). Remove the power connection and all the screws that hold the board. Remove the shield by pressing on the pins. Now, position the PCB in such a way that you are able to see the Adapter pins and the pins of the CPU on the bottom of the PCB. (The side that has no components on it.) What you want to test in this part, is the continuity between the Adapter pins and the CPU pins. Yes, to see if your soldering is good. Touch one lead of your continuity checker on pin #1 of the Adapter. Touch the other lead to pin #1 of the CPU on the bottom side. There should be continuity between the two. If not, check your soldering for that pin. Next, do the same with pin #2 right through to pin #40. When all is well, replace the shield and reattach the PCB to the case. Do not forget the power wire. Make a visual test to see that nothing is touching.

8 - The final test for the Adapter section is to plug in the computer and turn it on. If you do not get your regular screen, turn it off right away and go back to step #5. Check your computer with several pieces of software and all your peripherals plugged in. When you are satisfied that all is working well, continue the installation.

Plugging in the Boards.

You have now successfully installed the Adapter and are ready to continue. At this point, you will be unplugging and plugging the power to your computer many times. Please do this every time you are told, because if you plug or unplug a board while the power is on, you may damage the board, your computer or both. Unplug everything to your computer.

9 - Locate pin #1 of the small board. Insert the small board into the CPU Adapter making sure that pin #1 of the small board enters pin #1 of the Adapter. Rock the board slowly and lightly, back and forth until the connector of the board reaches the bottom of the Adapter. Never press hard onto the board to get it in! This may break the solder connections of the Adapter.

10 - Plug in the computer and turn it on. If you do not get your regular screen, turn it off right away, remove the small board and go back to step #5.

Installing the Special Memory Board.

You are now well on your way to a 1 Meg COCO. The next few steps involves installing and testing the Special Memory board. Be patient, its almost done.

11 - Unplug the computer and remove your regular 512K memory board. If you bought a 0K Special Memory board insert 16 - 41256 memory chips into the board. Make sure that pin #1 of the chips are inserted into pin #1 of the sockets. Insert the Special Memory board into the memory connectors of the COCO. Make sure that all the pins are inserted correctly. Insert the 1 pin wire of the Special Memory board into pin #3 of J2 of the small board. Pin #1 of J2 is next to the number '2'. Insert the power adapter (9 Volts DC @ 1 Amp) connector into J5 of the Special Memory board.

12 - Plug in the computer and turn it on. If you do not see your regular screen, turn it off right away, unplug everything and remove the Special Memory board. Replace your regular memory board and go to step #10. Run the memory 'MEMTEST' program supplied with this kit. At this point, you should get 512K of memory.

13 - Unplug the computer and insert your regular 512K memory board on top of the Special Memory board. Make sure that all the pins are connected properly. Plug in the computer and turn it on. If you do not get your regular screen, turn if off right away, remove your regular memory board and go to step #12. Turn the computer off and change the wire of the Special Memory board from pin #3 to pin #1 of J2. Run the memory test program again. This will test the regular memory board only. Again you will only get 512K of memory.

14 - Turn the computer off and change the wire of the Special Memory board from pin #1 to pin #2 of J2. Again, run the memory test program. If all is well, you now have 1 Meg of memory! Congratulations!

15 - Run the power adapter wire where it will not be in the

way. You may want to cut a notch in the case to pass the wire. Depending on which regular memory board you have, you may have to trim the long pins in order to properly close the cover. After the pins are cut, run the memory test again. Close up the computer.

Technical Information.

The COCO III uses a chip called the GIME. The following discussion presumes that you have a good knowledge of the GIME chip, its functions and its memory map. As you know, the GIME has 16 DAT (Dynamic Address Translation) registers. Any of these 16 registers can contain a number from 0 to 63. This number, 0 to 63, is known as a Ram Block. Each of these blocks consists of 8K of Ram giving you a total of 512K of memory. In a HEX value, 0 to 63 is represented by 6 bits.

What the DISTO Memory Upgrade does, is add 1 more bit to the Ram Block. The value that can be entered into the DAT Registers is now 0 to 127. The locations are exactly the same as the GIME DAT registers, from $FFA0 to $FFAF. But, unlike the GIME's DAT registers, you cannot read back the extra bits. This is no problem, since OS9 Level II does not need to read them. It is up to the software using the DAT registers to keep track of what is in each register.

Since there is no way for the GIME chip to know that extra memory exists, it can not put any video in that area. It also can not cross any 512K boundary with video. But, there is a circuit that allows you to move the video completely into other 512K banks. This is done by adding a register into an unused memory of the GIME chip. This location is at $FF9B. On power-up or reset, bit #0 of this location is set to 0. Writing a 1 to that location will force the video to come from

the second 512K bank. Again, this is a write only bit. It is up to the software to keep track of which bank the video is in.

If the Screen Start Address Register is set close enough to the end of a 512K boundary, the video will scan the beginning of the same 512K boundary and not cross over into the next 512K block. To avoid this, make sure that the start of video is in the proper location.

OS9 Operation

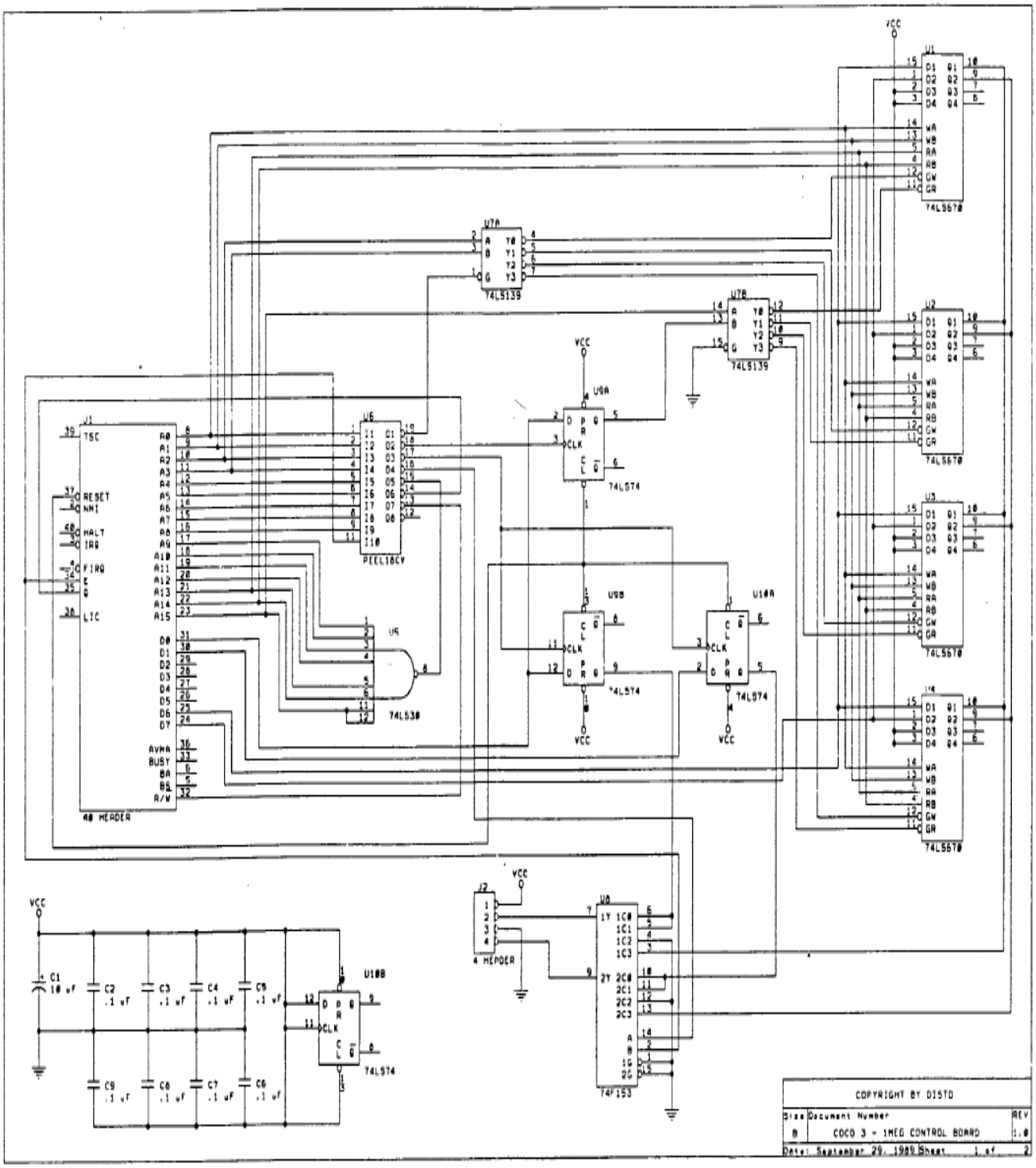
Complete OS9 drivers are included with this. When properly installed, the OS9 operating system will recognize a full 1 - MEG of memory. There are no restrictions on how to use this memory. All programs should be able to make use of full memory. Complete instructions on how to install these drivers are included in a README file on the OS9 portion of the included disk. There may be more than one README file, if so, read all of them before starting. The instructions will require that you to make a new Boot disk. When making a new Boot disk, remember to start with only original Tandy modules. Then, when you get the 1 - MEG working properly, you can start adding and modifying modules to customize your system to your needs. All OS9 drivers and patches are written and copyrighted by Kevin Darling unless other wise noted.

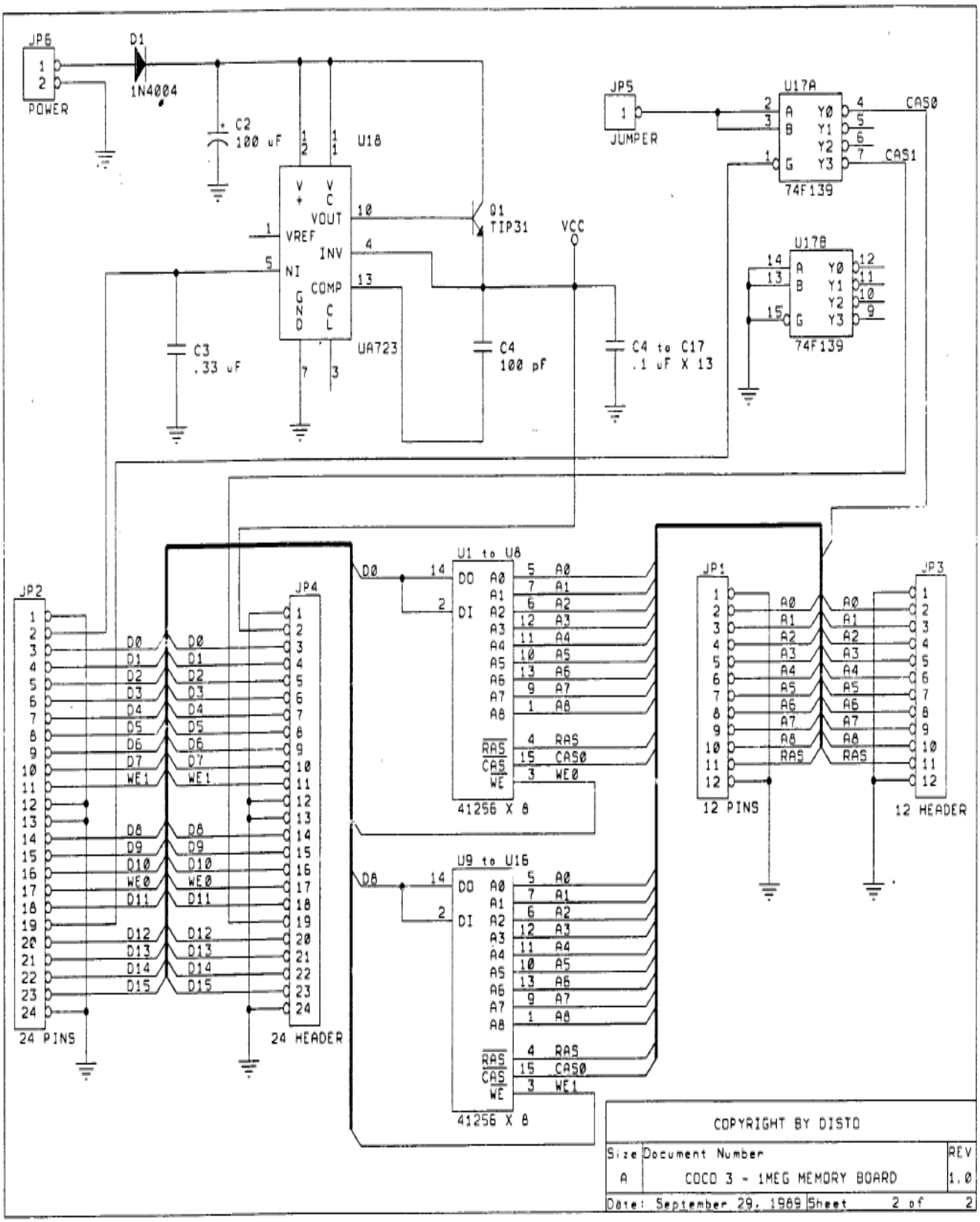
Credits:

The DISTO 1 Meg Upgrade Kit and all its documentation are conceived and designed by Tony DiStefano. The DISTO 1 MEG Upgrade Kit is manufactured and distributed by,

CRC Computers Inc.
11 Boul. Des Laurentides
Laval, Quebec
Canada H7G 2S3
1 (514) 967-0195

1-Meg Upgrade Schematics:




JP6
POWER
D1
1N4004
C2
100 uF
U18
UA723
Q1
TIP31
VCC
C3
.33 uF
C4
100 pF
C4 to C17
.1 uF X 13
JP5
JUMPER
U17A
74F139
CAS0
CAS1
U17B
74F139
JP2
24 PINS
JP4
24 HEADER
U1 to U8
41256 X 8
U9 to U16
41256 X 8
JP1
12 PINS
JP3
12 HEADER
COPYRIGHT BY DISTO
Size A
Document Number
COCO 3 - 1MEG MEMORY BOARD
REV 1.0
Date: September 29, 1989
Sheet 2 of 2

This is a rough draft OpenOffice.org Writer ".odt" and/or Adobe Acrobat ".pdf" transcript of Tony Distefano's original document. I found and downloaded the document from the maltedmedia ftp server, and so I assume that permission has been granted for the document to be posted there. I have corrected many errors in the original document and extended it somewhat. Please note that this rough draft still contains errors, and I will continue to refine it as I have time. Any input would be appreciated: sales@gimechip.com – Also, does anyone know how I can contact Tony?

Thanks – John Eric Turner and his Father