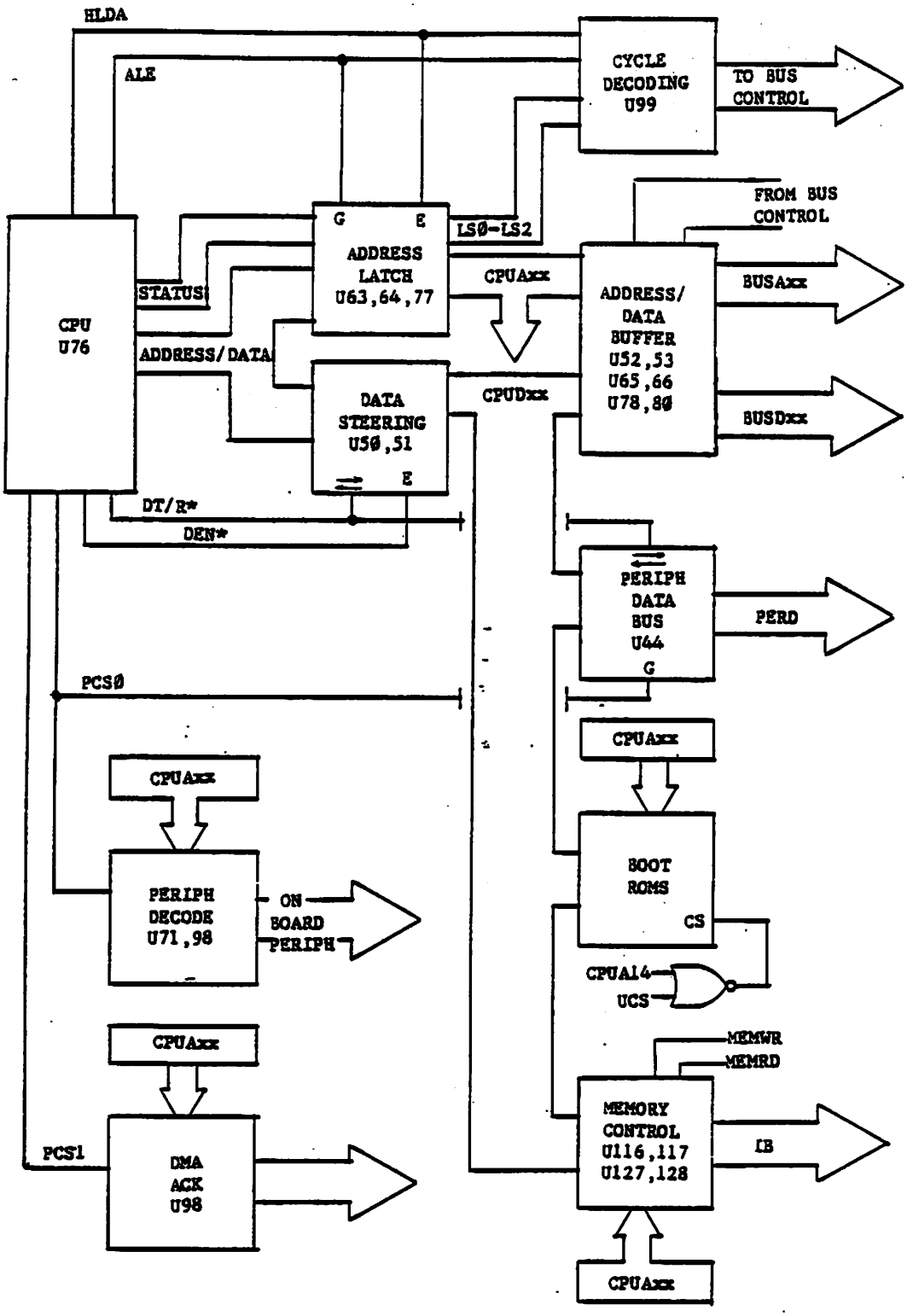
MODEL 2ØØØ BUS DECODING BLOCK DIAGRAM



HANDOUT - MODEL 2ØØØ BUS MAP
GDB - Ø2/14/84

The Model 2ØØØ Bus Interface section includes the ICs necessary to produce and control the iAPX 186 (8Ø186) processor bus.

There are four main buses:

CPU bus System bus Peripheral bus Internal Memory bus

The 8Ø186 processor is supplied in a 68 contact leadless chip. To provide its many functions with a limited number of pins, the 8Ø186 uses a MULTIPLEXED address and data bus. This means that the address and data lines share the same contacts on the processor. Some means of separating the two must be provided externally to the processor.

U63, U64, and U77 (schematic pg 2 of 13) are the ADDRESS DEMULTIPLEXERS. Their job is to latch the address when it becomes available. U5Ø and U51 are the DATA GATES. The data bus is not latched, but is simply enabled and steered as necessary.

The trick is knowing when the bus contains ADDRESSES and when the bus contains DATA. The only thing that knows for sure is the CPU itself. It supplies us with three timing signals - DEN*, DT/R*, and ALE.

ALE is the Address Latch Enable signal. This signal is timed by the processor such that the FALLING edge of ALE indicates that a valid address exists on the lines. ALE is buffered by part of U1Ø5 to become a new signal, CPUALE. It is also buffered by part of U78 to become BUSALE which appears on the motherboard. There is no difference between ALE, CPUALE, and BUSALE except a slight and hopefully negligible gate delay.

The falling edge of CPUALE is used to latch the address lines into the address latches, U63, U64, and U67. Note that only the lower 16 address lines (AØ-A15) are actually multiplexed. Address lines A16 - A19 and the processor status lines SØ - S2 are merely latched by the falling edge of CPUALE. No demultiplexing is actually necessary.

The address demultiplexers (U63, U64, U77) are also effected by CPUHLDA. This signal is identical to HLDA from the processor, except it is buffered by part of U1Ø5. CPUHLDA is produced by the processor in response to a HOLD signal received from another unit. CPUHLDA is used to float (tri-state) the address bus multiplexers while the other device uses the bus.

U5Ø and U51 are the DATA STEERING GATES. Their first job is to properly direct the flow of the data bus - to the processor or from the processor. This is controlled by the signal DT/R* (Data Transmit/Receive Not). If DT/R* is HIGH the gates are placed in an OUTPUT (transmit) mode. At this time the processor is attempting to place information on the data bus during a WRITE cycle. If DT/R* is LOW the gates are placed in a INPUT (receive) mode while the processor is attempting to READ information.

The processor must also make sure that the data bus is not enabled while address information is present. This is the primary function for the DEN* (Data ENable Not) signal. If DEN* is LOW the data bus buffers (U5Ø, U51) are enabled and data passes through unimpeded. If DEN* iS HIGH the data buffers are floated.

Note that DT/R* is buffered by part of U1Ø5 to become CPUDT/R*, and DEN* is buffered by part of U1Ø5 to become CPUDEN*.

There are three STATUS LINES - SØ, S1, and S2 (U79 pins 52, 53, 54). These lines are latched by U77 to become LSØ, LS1, and LS2. These lines are a direct indication of what the processor is doing at any given moment. There are five possible states the processor may be in:

1) Memory I/O 2) Peripheral I/O 3) Interrupt acknowledge
4) Halted 5) Passive

These states are reflected in the status lines as follows:

S2	S1	SØ	STATUS	
1	Ø	Ø	Instruction fetch cycle	
1	Ø	1	Memory READ cycle	-memory I/O
1	1	Ø	Memory WRITE cycle	
Ø	Ø	1	Peripheral READ cycle	
Ø	1	Ø	Peripheral WRITE cycle	-peripheral I/O
Ø	Ø	Ø	Interrupt Acknowledge	
Ø	1	1	Halt	
1	1	1	Passive	

You'll notice that generally the S2 signal can be used to indicate a memory cycle While the SØ and S1 lines are used to indicate READ or WRITE cycles.

The three status lines are decode by a 1-of-8 decoder IC, U99. Note that U99 is DISABLED by the CPUHLDA during a hold acknowledge time and by CPUALE while the address lines are being latched. HI3 is simply a pull-up resistor to +5 volts.

The processor also produces RD* (READ NOT) and WR* (WRITE NOT). These signals are combined with the decoded outputs of U99 to produce distinct READ and WRITE for the peripheral and memory cycles. The resulting signals are CPUIOR* (CPU I/O READ NOT), CPUIOW* (CPU I/O WRITE NOT*), CPUMW* (CPU MEMORY WRITE NOT), and CPUMR* (CPU MEMORY READ NOT). Both MR* (MEMORY READ NOT) and MRF* (INSTRUCTION FETCH NOT) are combined with RD* to produce CPUMW*.

Once the address and data lines are demultiplexed and latched they are collectively called the CPU BUS. This is as close as any device can get to the CPU itself. The address lines branch off to all parts of the main logic PC board without further buffers or gates. However, only the BOOT ROMS are connected directly to the data bus. All other devices are buffered from the CPU data bus in some manner.

The lower eight data lines (CPUDØ - CPUD7) are steered through U44 (pg 11 of 13) to become the PERIPHERAL DATA BUS (PERDØ - PERD7). This bus handles the 8-bit peripheral devices. Included are the FDC (Floppy Disk Controller), the interrupt controllers (pg 7 of 13), the line printer, configuration, and keyboard ports (pg 12 of 13), and serial I/O and timer ports (pg 13 of 13). The direction of U44 is controlled by CPUDT/R*. The chip is enabled by activity on the INTAK*, BUSDMACKØ* (BUS DMA ACKNOWLEDGE Ø NOT), or PCSØ* (PERIPHERAL CHIP SELECT Ø NOT) lines.

The complete 16-bit CPU data bus is sent to the memory control section (pg 1Ø of 13). The CPU data bus is gated by U117 and U128 onto the Internal Memory bus (IBØØ - IB15) during a memory write. The memory data is latched onto the CPU data bus by U116 and U127 during a memory read.

The entire CPU data, address, and control bus is gated by U52, U53, U65, U66, U78, and U8Ø to become the main system bus, indicated by the BUS--- in the name of the signal. The System bus runs mainly to the video generation section of the main logic PC board (pg4, 5, 6 of 13) but also becomes the main signals on the motherboard expansion connector, J5.

The various buses appearing in the 2ØØØ system must be tightly controlled. Devices using the HOLD line, and those using Direct Memory Addressing (DMA) can seize the bus for their own uses. Of course, two devices must NEVER control the bus at the same time.

In the case of DMA request the processor handles the bus control since the DMA controllers are built into the CPU itself. In the case of an external HOLD, the device producing the HOLD will control the bus. The only area on the main PC board capable of producing a HOLD is the video section, producing signal VIDHOLD. Other devices that plug into the motherboard may generate a HOLD (BUSHOLD*).

In the case of a HOLD the bus is handled by the HOLDing device through U79, U1Ø2, and U1Ø3. The important signals produced by PALs U1Ø2 and U1Ø3 are HOLD to the CPU, BUSADIR to U65 and U66, BUSAEN* to U65, U66, and U8Ø, BUSDEN* to U52 and U53, BUSDT/R* to U52 and U53, BUSHLDA* hold acknowledge to the bus, and VIDHLDA* hold acknowledge to the video section.

All computer systems must have enough "smarts" to be able to begin operating from virtually nothing. Even all RAM based systems like the Model II must contain enough non-volatile program to allow the system to load and run a more complete operating system without serious intervention of the user.

This non-volatile program is called a BOOTSTRAP and is contained in the BOOT ROM. For a boot ROM to be useful we need to know only one item - where in memory does the processor begin processing after a power up or reset? This address is defined by the manufacturer of the CPU. Once this address is known the bootstrap program can be written such that it will run at this memory location. The hardware is designed such that after a power up or reset the CPU will always find this program inside the ROM.

The 8Ø186 is defined to begin processing at address FFFFØH after power on or reset. The 8Ø186 also supplies a special signal called UPPER CHIP SELECT NOT (UCS*) on pin 34 of the chip package. UCS* is active any time an address in the upper memory area is accessed.

Of course, this upper memory area must be defined. After power on or reset the lower limit of the upper memory area is defined as FFCØØH. A little mathematics will show that this is the upper 1K of the total 1 megabytes of memory. Any address in the range of FFCØØH to FFFFFH will produce UCS*. This will set us up as a possible 1K of memory space available for the boot ROM.

We have already determined the the 8Ø186 processor starts processing at address FFFFØH. This falls within the 1K upper memory limit, so UCS* is active (LOW).

Referring to page 7 of 13 in the Model 2ØØØ schematics, we see that UCS* is gated with CPUALE by U133. This will assure that BOOT*, the ROM chip selects, will NOT be active when the address lines may be changing. BOOT* is also a function of the activity on CPUA14. BOOT* will be produced only if CPUA14 is HIGH, locking us into a 16K segment of the boot ROM.

Note that the 8Ø186 starts processing at FFFFØH, only 8 words below the top of memory. This would appear to be a problem, but remember that UCS* is active for at least the upper 1K of memory. The lower limit of the UCS* signal can be changed under software control. In fact, it may be expanded up to 256K bytes. The Model 2ØØØ boot ROMS are only 64K bytes, and it is doubtful that all 64K is used.

The 8Ø186 also produce a signal Lower Chip Select Not (LCS*) and four Middle Chip Select Not (MCSØ* - MCS3*). These signal serve the same purpose as UCS*, but for different memory areas. LCS* would be used to define a chip or chips in the lower memory area from addresses ØØØØØH up to 3FFFFH, but this signal is unused in the Model 2ØØØ. By definition, LCS* will not be active until its upper limit is programmed. Since LCS* is unused, its upper limit is never programmed, effectively disabling it.

Of the four MCS* signals, MCS2* and MCS3* are presently unused, but run to J4, the Math Co-processor connector, and are probably used on that board. MCSØ* and MCS1* are buffered by U81 (pg2 of 13) to become CPUMCSØ* and CPUMCS1*. These two signals are then buffered onto the system bus by U8Ø to become BUSMCSØ* and BUSMCS1*.

Like LCS* and UCS*, the MCS* signal can have their limits set by software. The TOTAL area covered by the MCS* lines can range from a minimum of 8K to a maximum of 512K. With in these limits the area is divided equally among the four MCS* lines. Therefore, if the area size is 8K, each MCS* line will define a 2K block within the 8K. If the area is 512K each MCS* line defines a 128K block. There are several limitations on the area size and where it may start. The most sever of these is that the limits of UCS*, MCS*, and LCS* may not overlap.

Since the limits for MCS* are programmable they may be reprogrammed to point almost anywhere (within their limitations) at will. One of the uses of MCSØ* and MCS1* are to set RAM memory limits. CPUMCSØ* and CPUMCS1* can be found in the memory timing circuitry on page 9 of 13.

MODEL 2ØØØ PERIPHERAL SELECTS

Like the memory select signals LCS, MCS, and UCS, the 8Ø186 CPU also supplies a set of peripheral chip select signals, PCSØ* - PCS6*. These signals can be used to decode various areas of memory for device I/O function. Like the memory signals, the peripheral select signals are totally defined by the operating system.

Like the memory select lines, the peripheral select lines are defined by an address limit which may be set under software control. Once the limit is set each peripheral select line (PCSØ - PCS6) defines a 128-byte block of memory. Using the system address lines these blocks of memory can be further broken down into small blocks as the specific device requires.

There seven PCS lines available for use.

PCSØ* - General I/O ports
PCS1* - DMA Acknowledge ports
PCS2* - 9ØØ7 CRTC ports
PCS3* - BUSPCS3*
PCS4* - BUSPCS4*
PCS5* - BUSPCS5*
PCS6* - unused

PCSØ* is further divided into smaller areas by combining it with address lines to produce the following ports:

PCSØPØA* - Speaker/Timer controls
PCSØPØB* - DMA multiplexer control register
FLDTC* - FDC DMA terminate
PCSØP1 - 8251A Serial I/O chip
PCSØP3 - 8272A Floppy Disk Controller
PCSØP4 - 8253 Timer
PCSØP5 - 8255 Parallel I/O
PCSØP6 - 8259A Interrupt Controller Ø
PCSØP7 - 8259A Interrupt Controller 1

The 8Ø186 contains 5 internal sources of hardware interrupts. These are:

Ø1) Timer Ø
Ø2) Timer 1
Ø3) Timer 2
Ø4) DMA channel Ø
Ø5) DMA channel 1

The 8Ø186 also has provisions for up to 5 external interrupts. Two of these interrupts are programmable as interrupt acknowledge outputs. Of course, the 8Ø186 also contains provisions for properly handling and prioritizing all ten interrupt sources.

Of the five external interrupt pins, one is a Non-Maskable Interrupt, or NMI. One use of the NMI is to signal that the power supply is not functioning properly. This is indicated by the ACLO* signal from JlØ, the power supply connector.

Of the remaining four interrupt pins, only two, INTØ and INTl, are programmed as interrupt signal input pins. The remaining two, INTAØ* and INTAl*, are the interrupt acknowledge pins. When the 8Ø186 is programmed in this manner the interrupt system is said to be in the CASCADE MODE.

However, there are 16 possible external interrupting devices available to the 8Ø186:

Ø1) Main logic memory parity error
Ø2) Memory/peripheral timeout
Ø3) Main logic serial I/O channel
Ø4) Second serial I/O channel option
Ø5) Main logic floppy disk controller
Ø6) Secondary floppy disk controller option
Ø7) Hard disk controller
Ø8) Second hard disk controller option
Ø9) Keyboard
1Ø) CTR 9ØØ7 video controller
11) Mouse
12) Line printer
13) Math co-processor option
14) Additional memory parity error
15) DMA programming error
16) Unused.

Handling these interrupts is done with two 8259A Interrupt Controller ICs, U42 and U43. These IC can prioritize or disable the interrupts, depending on how they are programed. The processor itself can also prioritize each device (INTØ or INTl).

When in the cascade mode the 8Ø186 expects the 8259 to supply an INTERRUPT VECTOR on request. The vector for each of the 16 interrupts is placed in its corresponding 8259A during the boot sequence of the software to be used. When an interrupt occurs the CPU acknowledges the interrupt and requests that the vector be placed on the peripheral data bus (PERDØ - PERD7). The processor then receives this vector and uses it as an index into a jump table to locate the entry point for the interrupt handling software.

The 8Ø186 CPU has provisions for two Direct Memory Access (DMA) channels built in. Using the circuitry built around U62 and U49, up to four DMA channels can be multiplexed onto the to CPU DMA channels.

Of the four DMA channels only two are used. Channel Ø comes from the Floppy Disk Controller (FDC). Channel 3 comes from the optional Hard Disk Controller card.

U49 is an 8-bit data latch. The lower 4 bits (Ø-3) are used as enable lines called DMAØEN - DMA3EN. These bits are used to enable the DMA request signals, BUSDMARQØ* - BUSDMARQ3*, onto the DMA input lines to the processor.

The upper 4 bits (4-7) of U49 are the DMA select bits called DMAØS - DMA3S. These bits are used to select which of the two DMA channels any particular DMA request will activate.

A DMA acknowledge is generated through software to the particular port requesting DMA. These signals are BUSDMAKØ* - BUSDMACK3*.

If a DMA programming error is produced (more than two enabled DMA channels routed to the same 8Ø186 channel) DMEINT16 is produced. This interrupt signals that a DMA error has occured.

OVERVIEW
iAPX 186 PROCESSOR

The Intel iAPX 186 processor (commonly referred to as the 8Ø186) is supplied in a 68 pin leadless chip carrier package. The 186 processor is upwards compatible with the 8Ø86 and 8Ø88 processors, and adds 1Ø new instruction types to those already available on the 8Ø86 and 8Ø88.

** General Purpose Registers **

The 186 is a true 16-bit processor in that it both accesses and operates on data and instructions in 16-bit words. It is also capable of 8-bit operation. The 186 contains a 16-bit Arithmetic Logic Unit (ALU) which is supported by fourteen (14) 16-bit registers.

Eight of these registers are general purpose registers used by the ALU. Four of the general purpose registers can be split into 8-bit pairs to allow operation on 8-bit data. These registers are called AX, BX, CX, and DX.

Two of the general purpose registers are Index Registers, commonly used as pointers into data areas. A 16-bit Base Register is also supplied. The function of the base register is almost identical to that of an index register, except that the index registers have the ability to be auto-incremented and auto-decremented. A 16-bit Stack Pointer is also supplied.

** Segment Registers **

Four of the 16-bit registers are assigned as 16-bit segment registers. These registers are used to effectively offset the actual address of the program, data area, or stack. By changing the values of the segment registers a program may be run in any area of the memory without need for position independent coding techniques.

During operation, the value of the segment register is shifted LEFT by four bits. Then, the base address of the program (or data) is added to the segment register value. This produces a 2Ø-bit absolute (physical) address to be accessed or executed. This result in a 2Ø-bit absolute address space, or 1 megabyte of addressable RAM area.

As an example, suppose a program is written such that it resides at a base address of 1ØØØH (hex). As well, suppose that the code segment register is set at 1ØØØH. When the program is run, the code segment register is shifted LEFT 4 bits to supply an address of 1ØØØØH. The program base address is added to this value, producing a physical address of 11ØØØH. This becomes the absolute address at which the program is running. The shifting and addition process is automatic, and need not concern the programmer that writes the specific application program.

It can be seen that, by manipulation of the segment registers, any specific program can be made to run virtually anywhere in memory without making any changes to the application program. This gives the 186 a strong basis for multi-processing, multi-user operation, or both simultaneously. Each process or user is assigned a unique, non-overlapping segment of memory. The master time-sharing control program simply keeps track of what is using each segment, and changes the segment registers at the appropriate time allow each process to run.

In addition to the code segment register, there is a data segment register, a stack segment register, and a fourth extra segment register. Due to the nature of the 186 machine code, it is possible to keep these four segments completely separate in memory. There is no direct or implied relation ship between these four segments. It is possible, and in some cases desirable, to keep these four segments in completely separate, distinct memory areas, so long as they do not interfere with other programs, data, or stack.

** Program Counter and Status Register **

The 186 also contains 16-bit instruction pointer or program counter. This register contains the base address of the next sequential instruction to be executed. Of course, interrupts and subroutines will effect the value found in this register. As explained above, the base value contained in the program counter is added to the shifted value of the segment register to determine the physical memory address within the 1 megabyte memory space.

The last of the fourteen registers is the status or flag register. This register is much like the flag registers found in other processors. The usual Zero Flag, Overflow Flag, Carry Flag, etc are present, but there are some significant additions.

There is a Single Step flag which controls a single step function. By setting this flag the processor can be made to step through a program one instruction at a time for debugging purposes.

There is a Direction Flag used by the processor when processing string information. By setting or resetting this flag the index registers can be made to auto increment or auto decrement after an instruction which accesses string data. This saves program execution time, as it is not necessary the the index register be incremented or decremented by the program itself.

** Bus Architecture **

The iAPX 186 contains all of the "usual" microprocessors features: address lines, data lines, clock generation, bus control circuitry, etc. with a few minor variations.

The 186 handles 20 address lines to address a total of 1 megabyte of address space. The physical address of an instruction or data has already been determined to be the result of left-shifting the segment register and adding the 16-bit base register involved to form the 20-bit absolute address.

In order to keep the number of pins manageable the 186 uses data/address line multiplexing. The 16 data input/output lines are multiplexed onto the same pins as the lower 16 bits of the address lines. Of course, the 186 also supplies the necessary timing signals to allow the address and data to be properly demultiplexed into and out of the processor.

The 186 contains logic to handle "slow" peripherals by the generation of wait states. Two "wait" lines are provided - ARDY and SRDY. They function identically, except that SRDY must be synchronized to the processor clock. ARDY requires no such synchronization. However, the 186 allows the number of wait states to be programmable, and the SRDY and ARDY lines may be ignored completely, in which case wait states may be automatically included or excluded under program control.

The processor also allows for bus sharing with other devices or processors. HOLD and HLDA are supplied for this purpose. If another device asserts the HOLD line the processor will relinquish the bus to the other device, but only after the processor has finished its current instruction. When it is finished the processor will assert the HLDA line to signal that the other device may control the bus. When the other device is finished it drops the HOLD line and the 186 continues processing.

** Additions **

At this point the iAPX 186 takes several radical digression from other processors. These digressions set the 186 far above many other processors.

** Prefetch Queue **

The 186 contains a 6-byte PREFETCH QUEUE. This section works almost independently of the processor, and strives to keep itself filled with the next 6 sequential bytes in memory. It can do this while the internal ALU is processing an instruction or data. When the ALU needs the next byte it is presented from the prefetch queue so that the ALU does not have to waste time waiting for the next byte to be fetched from memory. This results in a significant increase in instruction throughput.

However, make note that the prefetch queue fills itself with the next 6 SEQUENTIAL bytes. If the current instruction in the ALU results in non-sequential program flow (JUMP TO, JUMP TO SUBROUTINE, etc) the prefetch queue must be emptied and reloaded, and the ALU must now wait for the next instruction to be fetched from memory. However, the vast majority of program time is spent in sequential program flow, so the effects of prefetch queue dumping are minimal.

** Chip Selection **

The iAPX 186 also contains internal chip selection decoding circuitry. Three of these outputs determine three main memory areas - Lower Memory, Middle Memory, and Upper Memory. All three areas are programmable in size, and the upper and lower areas may be disabled.

The middle memory area is further decodes into four equal sized blocks. The starting point of the first block and the block sizes are programmable. All four blocks must be of the same size.

The 186 also decodes a separate peripheral memory area which is separate from and does not conflict with the system RAM memory area. The absolute address of the peripheral area is programmable. This area is effectively port addresses. The peripheral memory area is divided into seven 128-byte blocks, and one select output is provided for each block. Further address decoding must be accomplished through external circuitry.

** Interrupts **

The iAPX 186 is capable of directly handling 5 internal hardware interrupts, 5 external hardware interrupts, and 8 internal software interrupts, or traps. The hardware interrupts are prioritized by an internal interrupt controller.

Two of the five external interrupt pins can be programmed as interrupt acknowledge outputs, thereby allowing three external interrupts, two of which allow true handshaking capabilities. Using external circuitry, up to 16 external interrupts may be handled on the two available lines. The NMI interrupt has no handshaking capabilities, but is used as a "fatal error" interrupt, usually signifying that the power supply is about to die.

** Counters **

The iAPX 186 contains three internal counter circuits. All three counters may be clocked from the processor clock (8 MHz). Each counter is updated every four CPU clock cycles, therefore their maximum operation frequency is 2 MHz.

All three counters contain a 16-bit MAX COUNT register. When the counter reaches this count the output of the counter is activated. The output of counter 2 is not available externally to the processor. The counters may be used in this mode as a time delay device, and Counter 2 is often used a a DMA request timer.

Counters Ø and 1 each contain a second MAX COUNT timer, and they may be programmed to alternate between max count registers. The counter output directly reflects which max count register the counter is using. By proper programming of the max count registers the outputs of counter Ø or 1 may be used to generate waveforms of varying duty cycle.

Counters Ø and 1 may also be clocked by an external source which may be useful for event counting or real time applications. Additionally, each of the three timers may generate an internal hardware interrupt to the processor.

** DMA Controller **

The iAPX 186 contains a 2 channel DMA controller. Both channels are identical. Each channel may provide an interrupt to the processor.

Each DMA channel contains a 2Ø-bit (1 megabyte) SOURCE register, a 2Ø-bit DESTINATION register, and a 16-bit (64K) COUNT register, along with associated control registers.

The DMA channels can be programmed to operate in virtually any mode. Data may be transferred from memory location to memory location, from memory location to peripheral device, from peripheral device to memory location, or from peripheral device to peripheral device. Data may be transferred in 8-bit bytes or 16-bit words.

DMA may occur in a BYTE MODE where each byte (or word) of data is transferred and then the transfer is stopped to let some other device control the bus. If no other device needs the bus then another transfer takes place.

DMA may also occur in a BURST MODE where a block of data (whose size is set by the count register) is transferred. The DMA will not release the bus until the transfer is complete.

Data may also be transferred in a SYNCRONIZED mode. In this mode either the source or destination device may initiate the transfer. This allows devices like the FDC chip to accept or send data at its own pace. Data transfers between two like devices (memory to memory, for example) may be unsynchronized, running as fast as possible. The result is that DMA transfers can be accomplished at rates approaching 2 megabytes per second.

** Data Types **

One of the iAPX 186's most interesting features is its ability to handle different data types directly. All processors can handle standard 8-bit binary numbers, some can handle up to 16-bit binary numbers, and a few can handle Binary Coded Decimal (BCD). The 186 handles all of these, and more.

° Integer

A signed binary numeric value of either 8 or 16 bits. Integer operations assume 2's complement representation of the numbers involved.

° Ordinal

An unsigned binary value of either 8 or 16 bits.

° BCD

A byte representing the decimal digits Ø-9.

° Packed BCD

A byte representing two decimal digits Ø-9. Each nibble (4 bits) of the byte is one decimal digit.

° ASCII

A byte representing the alphanumeric and control codes specified by the ASCII standard.

° Strings

Strings are contiguous blocks of binary data. They may be up to 64K bytes in length. These strings should not be confused with the BASIC language STRING functions, as these are usually limited to to messages or ASCII data. The strings manipulated by the 186 have no limit on their individual byte values.

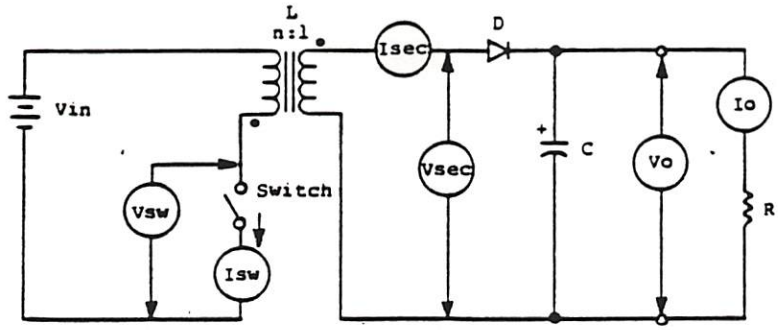


FIGURE 1. BASIC FLYBACK CONVERTER

95 - 135 VAC, nominal 120 VAC

MODEL 2ØØØ 95 WATT POWER SUPPLY

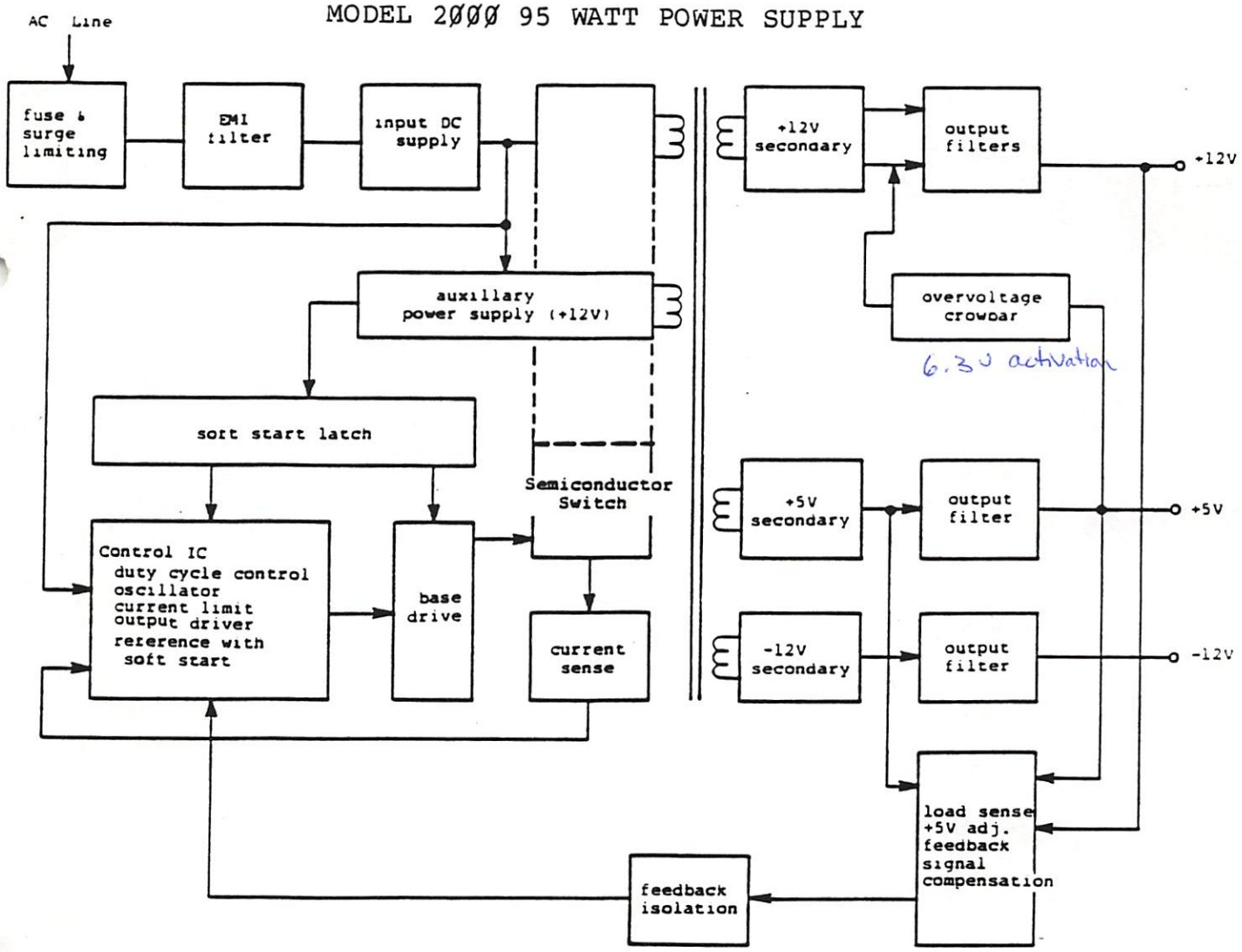
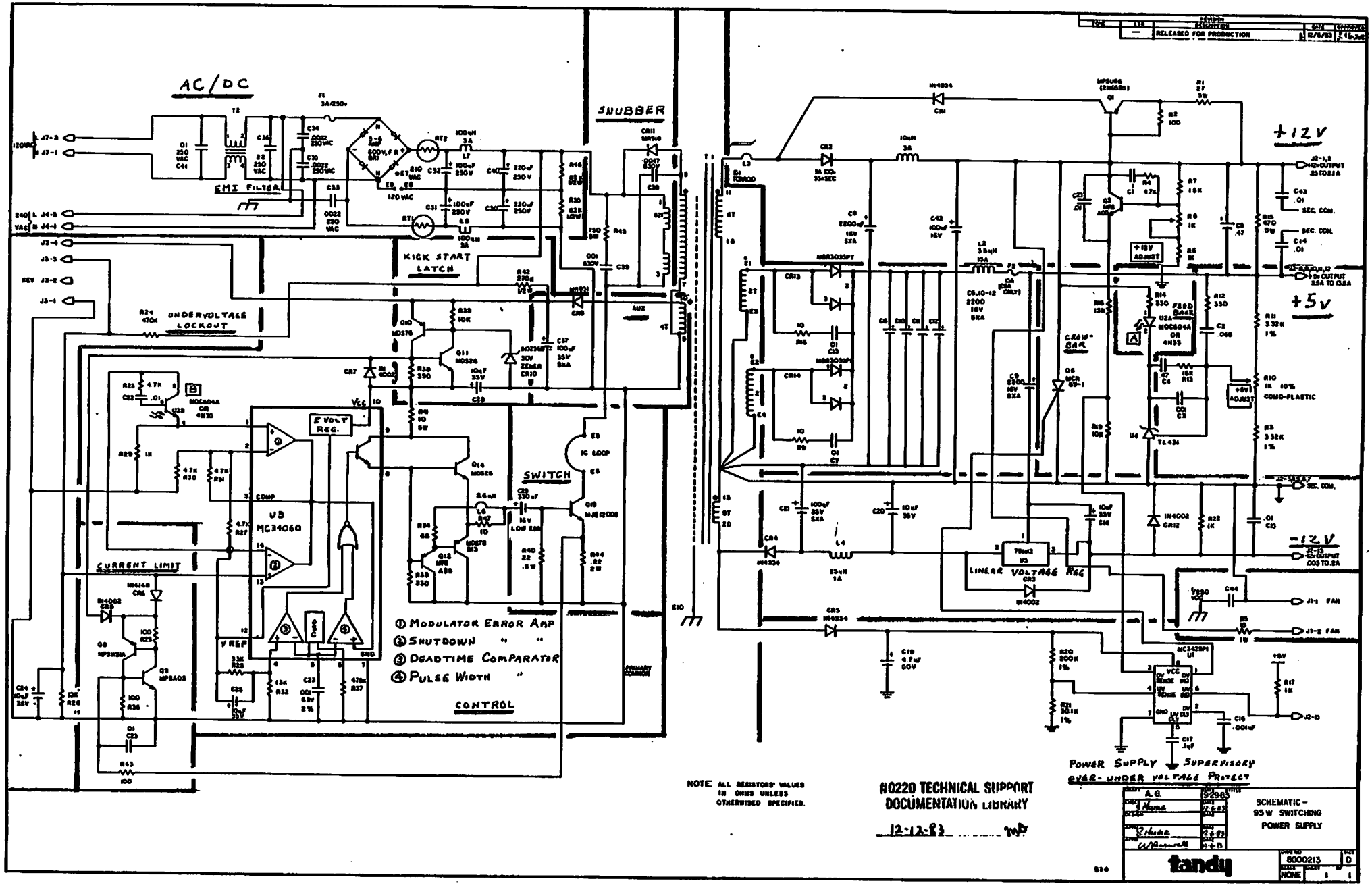


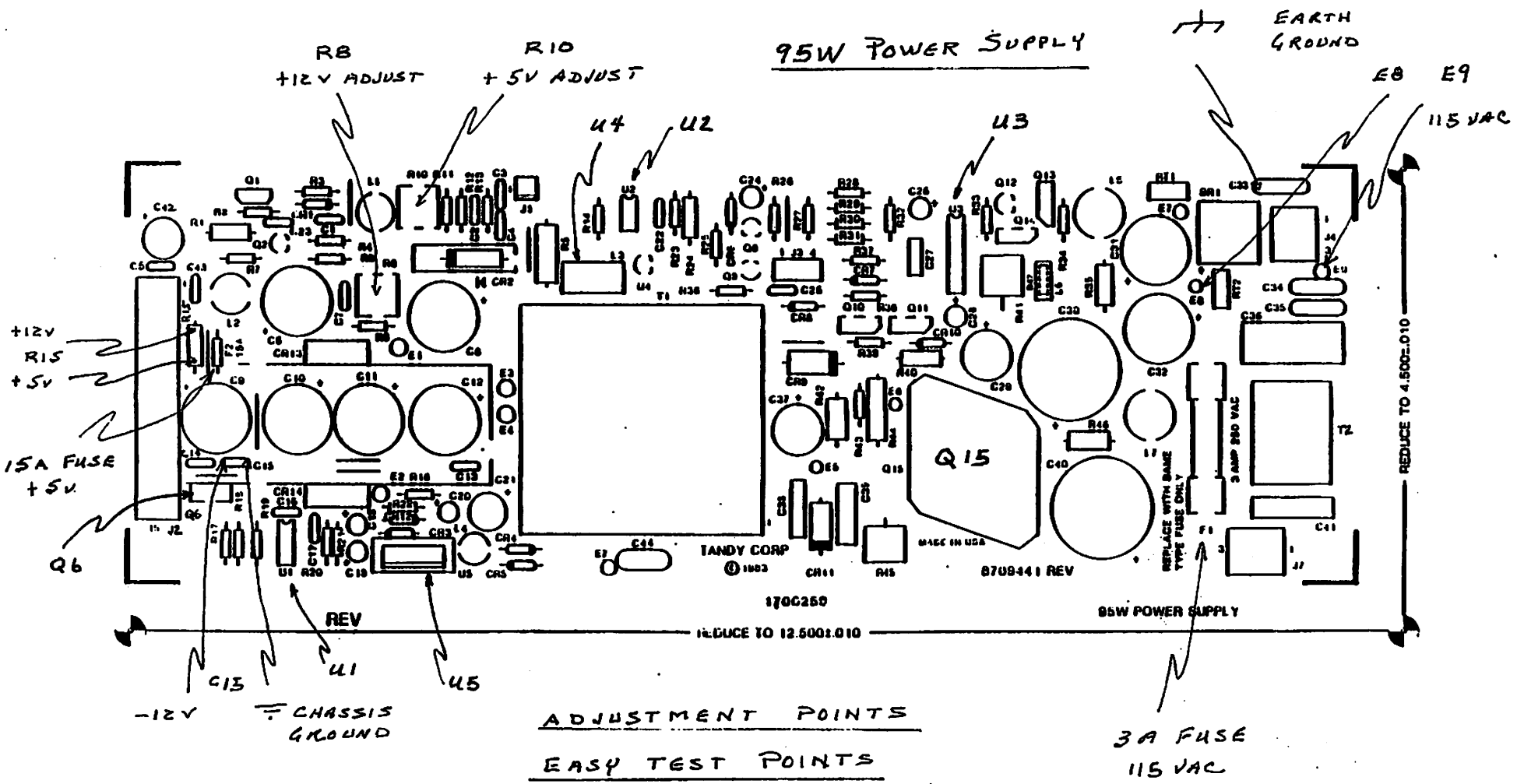
FIGURE 3. BLOCK DIAGRAM

hold up time - time where you still have voltage out with no switching going on

ours are ~~spec~~ specied for 10ms @ full load
18ms @ ½ load

increase primary (switch close) part of duty cycle to get more voltage out.


AC/DC
EMI FILTER
KICK START LATCH
UNDERVOLTAGE LOCKOUT
SNUBBER
SWITCH
5 VOLT REG.
CURRENT LIMIT
U8 MC34060
VREF
CONTROL
① MODULATOR ERROR AMP
② SHUTDOWN " "
③ DEADTIME COMPARATOR
④ PULSE WIDTH "
+12V
+5V
-12V
LINEAR VOLTAGE REG
POWER SUPPLY SUPERVISORY
OVER-UNDER VOLTAGE PROTECT
SCHEMATIC – 95W SWITCHING POWER SUPPLY
tandy
8000213
NONE
NOTE: ALL RESISTORS' VALUES IN OHMS UNLESS OTHERWISED SPECIFIED.
#0220 TECHNICAL SUPPORT DOCUMENTATION LIBRARY
12-12-83 TWD



ADJUSTMENT POINTS

EASY TEST POINTS