



MICROPROFESSOR MPF-1 PLUS USAGE NOTES

System ROM

Monitor Routines

0803	dfMBEL (BEEP)	Purpose.....: Call TONE to generate sound (BEEP) Input.....: None Output.....: None Registers...: AF, BC, DE, HL destroyed Labels.....: Monitor B)E)L)l
09B9	dfMCLR (CLEAR)	Purpose.....: Clear display buffer, set contents of DISP and OUTPTR to start address of display and input buffer respectively Input.....: None Output.....: (OUTPTR) <- INPBF (DISP) <- DISPBF Registers...: All saved Labels.....: Monitor C)L)ear) display buffer
07F6	dfMCLB (CLRBF)	Purpose.....: Clear display buffer, calls CLEAR, writes prompt at 1st position Input.....: None Output.....: (OUTPTR) <- INPBF+1 (DISP) <- DISPBF+2 IX <- DISPBF Registers...: AF, IX destroyed Labels.....: Monitor C)L)ear display B)uffer
0840	dfMCLD (CLRDSP)	Purpose.....: Clear display buffer Input.....: None Output.....: None Registers...: All saved Labels.....: Monitor C)L)ear D)isplay buffer
0AF4	dfMH2B (HEXBIN)	Purpose.....: Convert ASCII code to binary values until a non-hex digit is encountered Input.....: DE: points to HEX-ASCII string Output.....: HL: converted binary value Registers...: AF, BC, DE, HL destroyed Labels.....: Monitor H)ex 2) B)in
09CA	dfMSG (MSG)	Purpose.....: Convert ASCII code stored in input buffer to display patterns and put them to display buffer until a CR is encountered Input.....: HL: Address of message to process Output.....: HL: incremented by text length (OUTPTR) <- (OUTPTR)+ text length (DISP) <- (DISP)+2 * text length Registers...: AF HL destroyed Labels.....: Monitor M)eS)aG)e
0A89	dfMB4H (HEXX)	Purpose.....: Convert binary value in HL to ASCII codes and display pattern and store the ASCII code in (OUTPTR) and the display patterns in (DISP). Repeat this four times Input.....: HL: Binary value Output.....: (OUTPTR) <- (OUTPTR)+ 4 (DISP) <- (DISP)+2 * 4 Registers...: AF destroyed Labels.....: Monitor B)inary to 4) H)ex
0246	dfMSC (SCAN)	Purpose.....: Scan keyboard and display, wait for any key pressed, beeps on keypress. Do not call with running interrupts Input.....: IX: points to the buffer containing display patterns Output.....: A: ASCII code for the pressed key Registers...: AF, BC, HL, AF', BC' DE' HL' destroyed Labels.....: Monitor S)C)an

029B	dfMSC1 (SCAN1)	Purpose.....: Scan keyboard and display once, do not wait for key pressed Takes about 16mS to execute, crashes with running interrupts Input.....: IX: points to the buffer containing display patterns Output.....: A: position key code if a key was pressed CY: 1= no key pressed 0= key pressed, position key code in A Registers...: AF, BC, HL, AF', BC' DE' HL' destroyed Labels.....: M)onitor S)C)an1)
024D	dfMSC2 (SCAN2)	Purpose.....: Scan keyboard and display, wait for any key pressed Input.....: IX: points to the buffer containing display patterns Output.....: A: ASCII code for the pressed key Registers...: AF, BC, HL, AF', BC' DE' HL' destroyed Labels.....: M)onitor S)C)an2)
026C	dfMYM (KEYMAP)	Purpose.....: Convert key position code returned by SCAN1 to internal code Input.....: A: position code Output.....: A: internal code Registers...: AF, HL destroyed Labels.....: M)onitor K)eY)M)ap
0819	dfMRCK (RAMCHK)	Purpose.....: Check if given address is RAM Input.....: HL: Address to check Output.....: Z: 1= its RAM 0= it's ROM or non existsent Registers...: AF destroyed Labels.....: M)onitor R)am C)heck)
069A	dfMTW	Purpose.....: Write memory block to tape. Not an 'official' entry point. Calls DUMP after input of parameters. On parameter errors, monitor's error handler is called. Input.....: (dvMTNW): Filename, 4 characters (dvMTAF): From address (dvMTAT): To address (dvMTUD): 17 bytes user data, are saved in header block Registers...: None saved Labels.....: M)onitor T)ape W)rite
06D0	dfMTR	Purpose.....: Read memory block from tape. Not an 'official' entry point. Calls LOAD after input of filename. On load errors, monitor's error handler is called. Can only be aborted with RESET. Input.....: (dvMTNR): Filename, 4 characters Output.....: (dvMTAF): From address (dvMTAT): To address (dvMTUD): 17 bytes user data, are saved in header block Registers...: None saved Labels.....: M)onitor T)ape R)ead
08A8	dfMPCN (PTESTT)	Purpose.....: Check if printer board connected Output.....: Z: 0= not connected 1= connected Registers...: AF destroyed Labels.....: M)onitor P)rinter C)onN)ected

Monitor Routines on PRT-PPF

6A00	dfPSH (SHIFT)	Purpose.....: Drive the thermal head right, if at rightmost position, to the left Input.....: B: distance to move (45H ~ 1cm) Registers...: AF, B destroyed Labels.....: P)rinter S)H)ift
6A10	dfPLF (PLINEFD)	Purpose.....: Perform a line feed Registers...: AF, B destroyed Labels.....: P)rinter L)ine F)eed
6A30	dfPLF2 (PLINE)	Purpose.....: Perform two line feeds, works only if printer is ON Registers...: AF, B destroyed Labels.....: P)rinter L)ine F)eed 2)
6A40	dfPPLB (MPPRT)	Purpose.....: Print line buffer contents, terminated with CR. Starts on a new line Input.....: IX: address of line buffer Registers...: BC, AF', B' destroyed Labels.....: P)rinter P)rint L)ine B)uffer

Monitor ROM Patches

01EF	PRTMSG	<u>No printer init message. This just wastes paper</u> Original 01EF: CALL PRT_MPF CD 90 69 Prints ' *****MPF- I - PLUS*****' Patched 01EF: NOP, NOP, NOP 00 00 00 No print
01B0	PRTOFF	<u>Printer initially off</u> Original 01B0: LD HL, 0000 21 00 00 Sets both printer and beeper on. H is beep L is printer, where 0 means on, FF is off. Patched 01B0: LD HL, FFFF 21 FF FF This turns off the printer and beeper
0186	ROMMNU	<u>BASIC hotkey (ctrl-B) redirect to ROM Menu on PRT-MPF</u> Original 0186: LD A, (2000) 3A 00 20 ID byte of BASIC/FORTH ROM 0189: CP CD FE CD Check signature 018F: JP Z, 2000 CA 00 20 Jump to BASIC cold start 0192: JP 2020 C3 20 20 Jump to BASIC warm start Patched 0186: LD HL, (7000) 3A 00 70 ID byte of PRT-MPF ROM 0189: CP CD FE CD Check signature 018F: JP Z, 7000 CA 00 70 Jump to ROM Menu 0192: RET, NOP, NOP C9 00 00 for BASIC warm boot use ROM menu

Library ROM

The library resides in the PRT-MPF at 7700H and IOM-MPF at B700H. The library code contains jumps over I/O address areas which cause problems with interrupt operation due to improper I/O decoding. Must be assembled to addresses xx00, otherwise crashes will happen.

Interrupt Vectors

x700	IISVC0	CTC timer0
x702	IISVC1	CTC timer1
x704	IISVC1	CTC timer1
x706	IISVC3	CTC timer3
x708	IISVPA	PIO channel A
x70A	IISVPB	PIO channel B

Interrupt Handlers

x70C	dfLIS	Purpose.....: Set CPU interrupt mode, do not enable interrupts Input.....: A: interrupt mode to set (1 or 2, 0 is not available) HL: interrupt handler address (mode 1 only) Registers...: AF, I destroyed Labels.....: l)ibrary I)nterrupt S)etup
x71C	dfLIRMI	Purpose.....: Reset mode 1 interrupt vector to default Registers...: HL destroyed Labels.....: l)ibrary I)nterrupt R)eset M)ode 1) vector
x723	dfLID	Purpose.....: Check if interrupts are enabled, if so, disable them. Return enabled state, so the caller can enable them again if they were enabled. Output.....: CY: 0= ints are not enabled 1= ints were enabled before call Registers...: AF destroyed Labels.....: l)ibrary I)nterrupt D)isable
x72C	IIPA	Purpose.....: Interrupt handler PIO channel A, increments ints-occurred counter and calls user function if defined Registers...: AF, HL saved, the others are maintained by application Labels.....: l)ibrary I)nterrupt handler P)IO channel A)
x74B	IIPB	Purpose.....: Interrupt handler PIO channel B, increments ints-occurred counter and calls user function if defined Registers...: AF, HL saved, the others are maintained by application Labels.....: l)ibrary I)nterrupt handler P)IO channel B)
x76A	IIC0	Purpose.....: Interrupt handler CTC timer0. Calls user-supplied handler if not NULL, increments int occurred counter which can be checked and reset by application. Registers...: HL saved, the others are maintained by application Labels.....: l)ibrary I)nterrupt handler C)TC timer 0)
x789	IIC1	Purpose.....: Interrupt handler CTC timer1. Calls user-supplied handler if not NULL, increments int occurred counter which can be checked and reset by application. Registers...: HL saved, the others are maintained by application Labels.....: l)ibrary I)nterrupt handler C)TC timer 1)
x7A8	IIC2	Purpose.....: Interrupt handler CTC timer2. Calls user-supplied handler if not NULL, increments int occurred counter which can be checked and reset by application. This timer is also used for USART baudrate Registers...: HL saved, the others are maintained by application Labels.....: l)ibrary I)nterrupt handler C)TC timer 2)
x7C7	IIC3	Purpose.....: Interrupt handler CTC timer3. Calls user-supplied handler if not NULL, increments int occurred counter which can be checked and reset by application. Registers...: HL saved, the others are maintained by application Labels.....: l)ibrary I)nterrupt handler C)TC timer 3)

Interrupt Variables

FDC8	dvLCIC0	CTC timer0 interrupt counter, 1 byte
FDC9	dvLCIC1	CTC timer1 interrupt counter, 1 byte
FDCA	dvLCIC2	CTC timer2 interrupt counter, 1 byte
FDCB	dvLCIC3	CTC timer3 interrupt counter, 1 byte
FDCC	dvLPICA	PIO channel A interrupt counter, 1 byte
FDCD	dvLPICB	PIO channel B interrupt counter, 1 byte

8251 USART on IOM-MPF

x7E6	dfLUS	Purpose.....: Setup USART, call with disabled interrupts only Input.....: B: baudrate C: operating mode Registers...: All saved Labels.....: library USART Setup
x80D	dfLURTS	Purpose.....: Set/reset request to send handshake line Input.....: CY: 1= set RTS active 0= set RTS inactive Registers...: All saved Labels.....: library USART RTS)
x81E	dfLUDTR	Purpose.....: Set/reset data terminal ready handshake line Input.....: CY: 1= set DTR active 0= set DTR inactive Registers...: All saved Labels.....: library USART DTR)
x82F	dfLURB	Purpose.....: Check if a byte is received, if so, read it Output.....: A: received character if one was ready, 0 otherwise B: USART status Z: 0= data ready, returned in A 1= no data ready CY: 0= no error flags set 1= some errors were active, returned in B Registers...: AF, BC destroyed Labels.....: library USART Receive Byte
x84E	dfLUTB	Purpose.....: Check if USART transmitter empty, if so, send byte Input.....: A: byte to send Output.....: A: USART status CY: 0= transmitter was ready, byte sent 1= transmitter not empty, byte not sent Registers...: AF, BC destroyed Labels.....: library USART Transmit Byte

Z80 PIO on IOM-MPF

x85D	dfLPS	Purpose.....: Initialize one PIO channel, call with interrupts disabled Input.....: D: channel, 0=A, 1=B, or one of dcIPx E: operation mode, one of dcPMx B: interrupt control, one of dcPIxx A: input/output mask, 1=input, 0=output (mode 3 only) C: interrupt mask (mode 3 only) HL: address of interrupt handler CY: 1= setup PIO 0= disable channel interrupts, set mode 1 (all inputs) Output.....: Interrupts are disabled Registers...: AF destroyed Labels.....: library PIO Setup
------	-------	--

Z80 CTC on IOM-MPF

x8B9	dfLCS	Purpose..... Initialize or stop a CTC timer, call with interrupts disabled Input..... D: timer number, 0..3 E: control register contents B: time constant to use, one of hvC0xx or hvABxx HL: interrupt handler address, 0= no handler will be called CY: 1= setup timer 0= stop timer and interrupts Registers... AF destroyed Labels..... library C)TC S)etup
------	-------	---

DIP Switch on MPF-IOM

x8F4	dfLOSW	Purpose..... Get DIP switch status Input..... A: mask of switches to check for change Output..... A: unmasked switch status Z: 0= status has changed since last call 1= no change Registers... AF destroyed Labels..... library I)M S)W)itch
------	--------	--

Helper Routines

x910	dfLHDL	Purpose..... Delay for given number of 500 microseconds Execution times calculated for 1.7897725 Mhz clock Input..... DE=Number of 500 microseconds Registers... All saved Labels..... library H)elper D)eL)aY)
x922	dfLHMF	Purpose..... Fill memory area with constant Input..... A : Fill byte HL: Starting address BC: Number of bytes to write Registers... All saved Labels..... library H)elper M)emory F)ill
x935	dfLHHLDE	Purpose..... Compare HL and DE Input..... HL, DE to check Output..... SBC HL, DE flags Registers... F destroyed Labels..... library H)elper HL) DE)
x93E	dfLHDIV	Purpose..... Divide and get modulo Input..... HL: Dividend DE: Divisor Output..... HL: HL % DE, -1 if divisor = zero DE: HL / DE Registers... HL, DE Destroyed Labels..... library H)elper DIV)ide
x953	dfLHID	Purpose..... Check if character is decimal (0..9) Input..... A: Character to check Output..... CY: 0= It is decimal 1= It is not Registers... AF destroyed (F only) Labels..... library H)elper I)s D)ecimal
x95B	dfLHIH	Purpose..... Check if character is HEX (0..9, A..F) Input..... A: Character to check Output..... CY: 0= It is HEX 1= It is not Registers... AF destroyed (F only) Labels..... library H)elper I)s H)ex
x97D	dfLHSL	Purpose..... Get length of string by searching for either CR or 0 Input..... HL: String address Output..... A: Length of string Registers... AF destroyed Labels..... library H)elper S)tring L)ength

Converting Routines

0991	dfLOBH	Purpose.....: Convert binary to hex (0..FFFF), leading zeros always shown Input.....: DE: Value to convert HL: Where to store result C : Number of digits to convert (1..4) Output.....: (HL): Converted ASCII string Registers...: All saved Labels.....: l)ibrary c0)nvert B)inary to H)ex
x9D9	dfLOHB	Purpose.....: Convert HEX ASCII to binary (0..65535) Conversion stops at the 1st non-HEX or C gets zero Input.....: DE: start of ASCII digits C : Number of digits to convert (1..4) Output.....: HL: Binary value CY=1 if overflow or number of digits=0 Registers...: AF, HL destroyed Labels.....: l)ibrary c0)nvert H)ex to B)inary
xA11	dfLOBD	Purpose.....: Convert binary to decimal (0..65535) Input.....: DE: Value to convert HL: Where to store result C : Number of digits to convert (1..5) CY=1: leading zeros CY=0: leading blanks Output.....: (HL): Converted ASCII string Registers...: All saved Labels.....: l)ibrary c0)nvert B)inary to D)ecimal
xA92	dfLODB	Purpose.....: Convert decimal ASCII to binary (0..65535) Conversion stops at the 1st non-numeric or C gets zero Input.....: DE: start of ASCII digits C : Number of digits to convert (1..5) Output.....: HL: Binary value CY=1 if overflow or number of digits =0 Registers...: AF, HL destroyed Labels.....: l)ibrary c0)nvert D)ecimal to B)inary
xACC	dfLOUC	Purpose.....: Convert letter to uppercase Input.....: A: Letter to convert Output.....: A: Converted letter Registers...: AF destroyed Labels.....: l)ibrary c0)nvert to U)pperC)ase
xAD7	dfLOVC	Purpose.....: Validate character in A to see if it's printable. Does uppcase conversion first. Input.....: A: character to check/convert Output.....: A: converted character, space if non-printable CY: 1= non-printable Registers...: AF destroyed Labels.....: l)ibrary c0)nvert V)alidate C)haracter

Beeper/Display/Keyboard Routines

xAf1	dfLSBEEC	Purpose.....: Generate BEEP with parameters from system. Done here because the monitor routine is not IOM and PRT interrupt safe. Checks if system beep is enabled, if not, do nothing Registers...: All saved Labels.....: l)ibrary S)peaker B)E)E)P C)onditional
xAfB	dfLSBEEP	Purpose.....: Generate BEEP with parameters from system. Done here because the monitor routine is not IOM and PRT interrupt safe. Registers...: All saved Labels.....: l)ibrary S)peaker B)E)E)P)
xBOA	dfLSPK	Purpose.....: Generate BEEP. Done here because the ROM routine is not USART and PRT interrupt safe. Input.....: C: period = 2*(44+13*C) clock states HL: number of periods Registers...: All saved Labels.....: l)ibrary S)P)eaK)er

xB28	dfLKSOC	<p>Purpose.....: Read status of SHIFT and CONTROL keys. Those keys can be read without scanning the keyboard</p> <p>Output.....: Z: 1= SHIFT is pressed CY: 1= CONTROL is pressed</p> <p>Registers...: AF destroyed</p> <p>Labels.....: l)ibrary K)eyboard S)hift 0)r C)ontrol</p>
xB33	dfLKSAC	<p>Purpose.....: Check if shift and control are both pressed</p> <p>Output.....: CY: 1= both shift and control are pressed</p> <p>Registers...: AF destroyed</p> <p>Labels.....: l)ibrary K)eyboard S)hift A)nd C)ontrol</p>
xB48	dfLDT	<p>Purpose.....: Display a message for a given time</p> <p>Input.....: IX: address of message to display B: time to display the message. 030H ~ 1s</p> <p>Registers...: All saved</p> <p>Labels.....: l)ibrary D)isplay T)imed</p>
xB52	dfLDP	<p>Purpose.....: Display a pattern consisting of same characters. Uses direct port output, required in time-critical situations</p> <p>Input.....: A : character(s) to display DE: bitmap of digits 1-16 (E0=1, D7=16) a set bit turns on C : bitmap of digits 17-20 (B0=17, B3=20) a set bit turns on</p> <p>Registers...: All saved</p> <p>Labels.....: l)ibrary D)isplay P)attern</p>
OB8A	dfLDK	<p>Purpose.....: Display ASCII-string and scan the keyboard. CTRL-G toggles system beeper status</p> <p>Input.....: IX: address of string, must be 20 characters long</p> <p>Output.....: A: ASCII code of pressed key, if any, 0 if none Z: 0= a key was pressed CY: 1= shift and control were pressed</p> <p>Registers...: AF destroyed</p> <p>Labels.....: l)ibrary D)isplay K)eyboard</p>
xC5B	dfLDKR	<p>Purpose.....: Read a line of text from keyboard and display it. Text, decimal and HEX modes supported. Input width and position are configurable.</p> <p>Special keys:</p> <p>CR : end input, return with CY=0, Z=1</p> <p>left arrow : cursor left</p> <p>right arrow : cursor right</p> <p>ctrl-A : cursor to leftmost position</p> <p>ctrl-Z : cursor after last non-space character</p> <p>ctrl-K : toggle insert/overwrite mode</p> <p>ctrl-J : insert a space, shift buffer right</p> <p>ctrl-H : delete character, shift buffer left</p> <p>ctrl-U : restore initial buffer contents</p> <p>ctrl-Y : clear input field</p> <p>shift & control: abort, return with CY=1</p> <p>Input.....: IX: Address of text buffer If zero, it's a continued call, all other input params are ignored, they were stored at the initial call</p> <p>A: 0= text 1= decimal number 2= hex number</p> <p>B: Max number of characters to edit</p> <p>C: Position in text buffer where to place cursor</p> <p>CY: 1= clear text buffer initially 0= edit existing text in buffer</p> <p>Output.....: A: last pressed key, including invalid ones, 0 if none CY: 1= input was aborted with shift and control Z: 1= input was terminated with CR 0= either no input or something else than CR DE: If input terminated with CR and numeric input was active, the numeric value, zero otherwise</p> <p>Registers...: AF, DE destroyed</p> <p>Labels.....: l)ibrary D)isplay K)eyboard R)ead</p>

Library Setup

xF4A	dfLLS	Purpose.....: Initialize library variables. Should be called once at start of application Registers...: All saved Labels.....: l)ibrary S)etup
------	-------	--

Library Version

xF5B	dbLVMAJ	Li brary versi on, maj or part
xF5C	dbLVMIN	Li brary versi on, mi nor part

RAM Utilities

These utilities are not in ROM, they must be loaded either from Cassette or with Bin/HexRcv.

Library Test

This utility is for testing or debugging the library functions. It must be loaded either from Cassette or with RcvBin/RcvHex. All functions work with running interrupts. On startup a menu with the available functions shows up:

SPACE		Switch the menu lines
SHFT-CTRL		Terminate, back to monitor if in menu, back to menu otherwise
<u>D</u> -DLY	1UDLY	Delay calibration. Can only be leaved with RESET once started with another D. Toggles PIO PA pin in 10mS intervals. Use oscilloscope to calibrate by changing the value in ORG+013B/C. Results may be inaccurate if run with enabled interrupts (SW8). Shows 'D-START PA0=10MS' while waiting, display is dark while calibrating.
<u>C</u> -CNT		Shows interrupt counters as aaaa bbbb cccc dddd aaaa: CTC0 10mS as HEX bbbb: CTC1 20mS as HEX cccc: CTC3 30ms as decimal with leading zeroes dddd: PIO-A as decimal with leading spaces. PA7 must be jumpered to a PA0-5 output
<u>E</u> -EDT	1DKR	Textline editor, shown as 'TEXT 12345 ABDC' TEXT edits text 12345 edits a decimal number ABCD edits a hexadecimal number CR changes between input fields CTRL-P prints the line, with interrupts turned off
<u>S</u> -SER	1Uxx	Toggles UART echo mode. If enabled, echoes received characters back to sender. The green LED shows the enabled state (BEEP destroys it). Shows 'USART ECHO ON' or 'USART ECHO OFF' for a moment when toggled. Allows non-blocking transfer, the display flickers. The UART is setup for 4800,N,8,2 requires RTS/CTS handshake.
<u>V</u> -VER		Shows the version of the library stored in ROM for a second.
DIPSW 8		Enable/disable interrupts. On startup, they are disable, regardless the switch status. The interrupts are signaled on PIO-Pins: PA0: 10mS CTC timer 0, in callback PA1: 20mS CTC timer 1, in callback PA2: 30mS CTC timer 3, in callback PA3: 10mS CTC timer 0 /2, in main loop PA4: 20mS CTC timer 1 /16, in main loop PA5: 30mS CTC timer 3 /16, in main loop PA6: PIO-PA7 interrupt /3, in main loop PA7 is setup as interrupt input on rising edge. Connect PA0-5 to PA7 to trigger PIO-A interrupts at different rates.

Laufschrift

Display texts edited with the editor ED as a scrolling demo. Has it's own load from cassette, serial and floppy functions. The file format is compatible with the editor. It has two entry points, ORG+0 is cold boot, ORG+3 warm boot. If text is already in memory, display starts immediately, otherwise the load menu is displayed as with cold boot. The following function keys are available while text is displayed:

CTRL-L	<p>Display the load menu T) APE S) ER V) C1541 The menu is skipped if no MPF-IOM is connected, load from tape is executed in this case.</p> <p>T Load from tape. The program has it's own load routine which allows to load at a specified, not the saved-from address. The first file found is loaded, filename is ignored. Can be aborted with SHIFT-CONTROL as long as the loading has not started (no level change on EAR input).</p> <p>S Load from serial, 4800,N,8,1, can be aborted with SHIFT-CONTROL</p> <p>V Load from VC-1541 Floppy, if connected. VC1541 NOT PRESENT is shown otherwise. NAME then asks for the filename to load.</p> <p>Ctl-D Display floppy directory</p> <p>Ctl-F Display file info, start address and filetype</p> <p>Ctl-C Enter and send command to floppy</p>
CTRL-C	Toggle line-delete mode. If on, the displayed line is deleted before the new line scrolls in. If off (default), scrolling happens without deleting, the current text is scrolled out.
SPACE	Toggle pause mode. If on, the current line stays displayed, also shown by the green LED.
Arrow left/right	Change scroll speed, left=faster, right=slower
Arrow up/down	Change the time a line stays displayed, up=longer, down=shorter
SHIFT-CONTROL	Terminates program while displaying or exits load menu

ROM Utilities

Several Utilities are stored either in IOM-MPF ROM (B000- BFFF) or spare ROM U6 on PRT-MPF (7000- 7FFF). All utilities require the library.

Receive Binary

Used to download binary files to the MPF-1 Plus. It uses the UART on IOM-MPF with 4800,N,8,1, RTS/CTS handshake is supported. RcvBin has four entry points for different operating modes. Common to all modes is the cancel option with shift-ctrl, which aborts any action in progress, the transfer prompt '(((((' and the error messages

- <addr> VERIFY ERROR Written byte did not verify (ROM or no RAM area)
- <addr> SYS-RAM OVR System RAM (FCC0- FF00) overwrite attempt
- <addr> USART ERROR The 8251 USART signaled a receive error

The wait prompt '-----' signals waiting for data, '=====' waiting for system data

B000	RB binary	Binary mode <ul style="list-style-type: none"> • Start the program with <G>B000 or use the RomMenu, CTRL-B, R-RB. • The store address can be entered at the RB- FROM prompt as 4-digit HEX, editing is possible with left-arrow key. If no address or 0 is entered, F000 is assumed. The entered address must be in RAM, otherwise it is refused. • '-----' is shown, waiting for the first startbit • Start sending the binary file • '(((((' pattern is shown during the transfer. • At end of a successful transfer FROM xxxx TO xxxx is shown • Always exits to the monitor
B004	RB asm/edit	Assembler/Editor mode <ul style="list-style-type: none"> • Start the program with <G>B004 or use the RomMenu, CTRL-B, CTRL-R-A. • '=====' is shown, waiting for the first startbit • Start sending the assembly source file • '(((((' pattern is shown during the transfer • System data is received (sent by SendBin, FED9- FEED, contains pointers for Ed/Asm) • Source data is stored at address given when initializing the editor. Be sure to do it before receiving. File length and number of lines are calculated from received system data • At end of successful transfer quits to editor warm boot, otherwise to monitor. • Editor must have been cold booted since power-on, unpredictable results otherwise.
B008 45064	RB basic	Basic mode <ul style="list-style-type: none"> • Start the program with <G>B008, use the RomMenu, CTRL-B, CTRL-R-B or CALL 45064 from within Basic. • '=====' is shown, waiting for the first startbit • Start sending the basic source file • '(((((' pattern is shown during the transfer • System data is received (sent by SendBin, FED9- FEED, contains pointers for Basic) • Source data is stored at address given when initializing Basic. File length is calculated from received system data • At end of successful transfer or CALLED from Basic, quits to basic warm boot, otherwise to monitor. • Basic must have been cold booted since power-on, unpredictable results otherwise.
B00C	RB call	Call mode <ul style="list-style-type: none"> • The filetype must be in A on calling. If binary, startaddress is asked, if any other it must be written to system start location at FED9 • To start, call B00C • '-----' is shown for binary, '=====' for other types, waiting for the first startbit • Start sending the file • '(((((' pattern is shown during the transfer • Data is stored at address specified in FED9 • At end of successful transfer the last written address (or 0 if aborted) is stored in FED6 (tape end) so the caller knows the length of the received data • Always RETURNS to caller.

Send Binary

Used to upload binary or text files from the MPF-1 Plus. It uses the UART on IOM-MPF with 4800,N,8,1, RTS/CTS handshake is supported. SendBin has four entry points for different operating modes. Common to all modes is the cancel option with `shift-ctrl`, which aborts any action in progress, and the transfer prompt `''))))'`. The wait prompt `'-----'` signals waiting for sending data, `'====='` waiting for sending system data.

B1E8	SB binary	Binary mode <ul style="list-style-type: none"> Start the program with <code><G>B1E8</code> or use the RomMenu, CTRL-B, S-SB. The start- and end addresses must be entered at the SB-FROM xxxx TO xxxx prompt as 4-digit HEX each, editing is possible with left-arrow key. The TO address must higher than the FROM address, otherwise it is refused. <code>'-----'</code> is shown, waiting for the first two bytes sent (CTS must be active) Start receiving the binary file <code>''))))'</code> pattern is shown during the transfer. At end of a successful transfer SB-FROM xxxx TO xxxx is shown Always exits to the monitor
B1EC	SB asm/edit	Assembler/Editor mode <ul style="list-style-type: none"> Start the program with <code><G>B1EC</code> or use the RomMenu, CTRL-B, CTRL-S-A. <code>'====='</code> is shown, waiting for the first two bytes sent (CTS must be active) Start receiving the assembly source file <code>''))))'</code> pattern is shown during the transfer System data is sent (FED9-FEEC, contains pointers for Editor and Assembler) Editor text in the range specified in system data is sent At end of successful transfer SB-FROM xxxx TO xxxx is shown and quits to editor warm boot (same as CTRL-R in monitor), otherwise to monitor. Editor must have been cold booted since power-on, otherwise unpredictable results may occur.
B1F0 4552	SB basic	Basic mode <ul style="list-style-type: none"> Start the program with <code><G>B1F0</code>, use the RomMenu, CTRL-B, CTRL-S-B or CALL 4552 from within Basic. <code>'====='</code> is shown, waiting for the first two bytes sent (CTS must be active) Start receiving the basic source file <code>''))))'</code> pattern is shown during the transfer System data is sent (FED9-FEEC, contains pointers for Basic) Source data is sent from addresses in system data At end of successful transfer, SB-FROM xxxx TO xxxx is shown and quits to basic warm boot, otherwise to monitor. If CALLED from Basic, it always does a Basic warm boot. Basic must have been cold booted since power-on, otherwise unpredictable results may occur.
B1F4	SB call	Call mode <ul style="list-style-type: none"> The user program must write the start address to location FED9 (editor start), the end address to FEDB (editor end) To start, call B1F4 <code>'-----'</code> is shown for binary, <code>'====='</code> for other types, waiting until the first two bytes are sent. Start receiving the file <code>''))))'</code> pattern is shown during the transfer Data is sent from addresses specified in FED9 to FEDB Always RETURNS to caller.

Receive HEX

B360	RH	<p>Used to download Intel HEX files to the MPF-1 Plus. It uses the UART on IOM-MPF with 1200,N,8,1, supports RTS/CTS handshake.</p> <ul style="list-style-type: none"> • Start the program with <G>B360 or use the RomMenu, CTRL- B, H- RH. It can be terminated any time with the SHIFT- CONTROL key combination. • '-----' is shown, waiting for the first startbit • Start sending the HEX file. • '(((((' pattern blinks after every received HEX line. The code is stored at the address given in the HEX file and cannot be changed. • At end of a successful transfer RH- FROM xxxx TO xxxx is shown. Terminate the program with any key. • Error messages <ul style="list-style-type: none"> NO START MARK No start mark (:) found in HEX line LINE TOO LONG Received line contains more than 80 characters CHECKSUM ERROR Line checksum mismatch VERIFY ERROR Written byte does not verify (ROM or no RAM area) SYS- RAM OVR System RAM (FCC0- FF00) overwrite attempt USART ERROR The 8251 USART signaled a receive error
------	----	--

Stopwatch

B500	SW	<p>This is a simple stop watch, it uses the Z80 CTC on the IOM-MPF as time base and runs with interrupts.</p> <ul style="list-style-type: none"> • Start with <G>B500 or use the RomMenu, CTRL- B W- SW • 00: 00: 00- 00 appears. • Press SPACE to start/stop the stopwatch • Use ' R' key to reset the counters to zero • The 10mS interrupt is counted on PIOs PA port • Terminate at anytime with SHIFT- CONTROL
------	----	---

Commodore VC-1541 Floppy Drive

D800 55296	VC	<p>This program was adapted from an article in MC 04/84 which describes how to access a Commodore VC1541 from Z80-systems. It uses a hardware interface which connects the IEC lines to PIO port B on the IOM board.</p> <p>File Types</p> <p>The program uses different file types to select the mode of operation. The type byte is stored at the very first position in the file.</p> <ul style="list-style-type: none"> • bl nary Load and save operations take place in raw format, start and end addresses must be entered, no system data load or save. • Assembler Intended for MPFs Editor/Assembler. Saves and loads the system area from FED9 to FEED, containing the parameters for Editor and Assembler. Start- and end addresses are taken from this system area. On exit, enters Editor warm boot if a LOAD operation was done (same as CTRL- R from monitor) • Basic Intended for MPFs Basic. Saves and loads the system area from FED9 to FEED, containing the Basic parameters. Start- and end addresses are taken from this system area. On exit, enters Basic warm boot if a LOAD operation was done. If called from Basic, this type is automatically set. • Ed The editor ED uses this type. As it can also be used for assembler sources, the system area from FED9 to FEED, containing the parameters for the Assembler, are also saved and loaded. To be used in calling mode only. <p>Operation</p> <ul style="list-style-type: none"> • Start with <G>D800 or use the RomMenu CTRL- B <u>V</u>- VC • The menu is displayed, switch with SPACE between the two lines • Select the desired function, returns to menu when done. If something went wrong, VCs error channel is read and displayed. • Terminate with SHIFT- CONTROL. Depending on the file type used for load or save, it quits to monitor for bl nary, to Editor for Assembler or to Basic for Basic type. If started from within Basic, it always returns to there. <p>Menu Items</p> <ul style="list-style-type: none"> • <u>L</u>- LO Load a file, specify filename at the NAME prompt. Returns to menu on empty input. File type and load address are now read. If bl nary, the load address can be entered at the FROM prompt. If 0 or empty, the address stored in the file is shown, if not cancelled with shi ft- control , this address is used. All other file types do not ask for the load address, it is taken from the system area, specified when Editor or Basic were initialized (must have been done before loading). It is shown for confirmation anyway and loading can be cancelled with shi ft- control . While loading, memory is verified. On a write error, <addr> COMPARE ERROR is shown. Additionally, overwriting the system memory is checked, if it happens, <addr> SYS RAM OVR shows up. • <u>V</u>- VY Verify a file with memory contents. The stored saved-from address is used as starting address, independent of file type. On a mismatch, <addr> VERIFY ERROR is shown. • <u>S</u>- SA Save a memory range to diskette. Enter a filename at the NAME prompt (an existing file can be overwritten by prefixing the filename with @:), then the type menu I - B I N A- ASM B- BAS comes up. Select the wanted type. For bl nary, from and to addresses must be specified, all other types use the addresses stored in the system area which is saved prior to the real data. Care must be taken that the system area is valid, either Editor or Basic must have been run before. • <u>D</u>- DI Display the disk directory line-by-line. A ctrl - P prints the current item and all following to the MPF-printer (if connected). • <u>F</u>- FI Display file info, load address and file type (xxxx LOAD x TYPE) • <u>E</u>- ER Read and display the VC error channel. • <u>C</u>- CM Enter and send a command to the floppy. Can be used to format, validate, rename or copy files. Input length is restricted to 16 characters.
---------------	----	---

Using the VC-1541 Routines without Menu

The routines in the driver can also be used without the menu. This is done in Editor application described in the next chapter. Before load or save functions can be used, the start address must be written to FED4 (tape start) and the end address to FED6 (tape end).

D888 D88B	dfVCINI dfVCINIL	Purpose..... Initialize and check if device present. If used without the menu, this function must be called before any other. Output..... CY: 1= device not present Registers... None saved Labels..... VC 1541 I)N)I)tialize [N)o L)ibrary init]
D8AC	dfVCINID	Purpose..... Reset VC and de-initialize PIO Registers... None saved Labels..... VC 1541 I)N)I)tialize D)elete
DA4E	dfVCRST	Purpose..... Reset the VC1541 Output..... CY: 1= device did not respond Registers... A, DE destroyed Labels..... VC 1541 R)eS)eT)
DA98	dfVCE	Purpose..... Read and display error channel Input..... CY: 1= show the read data 0= just read, do not show Registers... None saved Labels..... VC 1541 read E)rror channel
DADC	dfVCC	Purpose..... Read command from keyboard and send it to IEC Registers... None saved Labels..... VC 1541 C)ommand
DAFE	dfVCL	Purpose..... Load file from disk Output..... (vVCLLA): Last written address, 0 if error CY: 1= file not found Registers... None saved Labels..... VC 1541 L)oad
DB0C	dfVCV	Purpose..... Read filename and verify file on disk Registers... None saved Labels..... VC 1541 V)erify
DB56	dfVCI	Purpose..... Display/Print directory Registers... None saved Labels..... VC 1541 dI)rectory
DB15	dfVCF	Purpose..... Show load address and type of a file Registers... None saved Labels..... VC 1541 F)ile info
DB20	dfVCS	Purpose..... Save a file on disk Input..... CY: 0= called by user program, it has set from/to addresses 1= called from driver menu B: File type if called by user program Registers... None saved Labels..... VC 1541 S)ave

Text Editor

7150	ED cold	<p>This is a simple text line editor with printing support, located in PRT-MPF spare ROM. Text is always stored as full lines regardless how many characters are present. Text buffer is located at E000- FCC0 (368 lines) if IOM is present or F000- FCC0 (166 lines) if no IOM is connected.</p> <p>For UART functions the parameters are 4800,N,8,1, RTS/CTS required.</p> <p>The editor can also be used for assembler sources, on exiting ED, it writes text buffer start / end addresses and line count to system area, so the assembler knows where to find the source code.</p> <p>Operation</p> <ul style="list-style-type: none"> • Start with <G>7150/3 or use the RomMenu, CTRL-B <u>E</u>- ED, terminate with SHIFT-CONTROL. Can be warm-started if the first text line consists of all printable characters, already present text is retained. • An empty line with a blinking cursor is shown. On warm start the last active line is shown. <p>Commands</p> <p>All commands can be cancelled with SHIFT-CONTROL</p> <p>Cursor movement</p> <ul style="list-style-type: none"> • up-arrow one line up • down-arrow one line down • left-arrow cursor to left • right-arrow cursor to right • ctrl-A cursor to first position • ctrl-Z cursor right, after 1st non-space <p>Navigation</p> <ul style="list-style-type: none"> • ctrl-T go to top of text • ctrl-B go to bottom of text • ctrl-O go to specified line <p>Manipulation</p> <ul style="list-style-type: none"> • ctrl-J insert a space at cursor position • ctrl-H delete a character • ctrl-I insert empty line at current position • ctrl-D delete current line • ctrl-C copy current line • ctrl-V paste copied line • ctrl-Y clear current line (fill with spaces) • ctrl-U restore initial line contents <p>Miscellaneous</p> <ul style="list-style-type: none"> • ctrl-F find a string • ctrl-N find next • ctrl-K toggle ins/ovr mode (cursor changes shape) • ctrl-Q show current line number, used and available lines • ctrl-S save menu • ctrl-L load menu • ctrl-P print range of lines • ctrl-G toggle key buzzer • shift-ctrl return to monitor, in queries quit to editor <p>In load/save menu</p> <ul style="list-style-type: none"> • T load/save text from/to tape, no filenames supported • S load/save using UART on IOM • V load/save to floppy with file type ED • ctrl-D display floppy directory • ctrl-F display file load address and type • ctrl-C enter and send command to floppy drive <p>In load menu only</p> <ul style="list-style-type: none"> • ctrl-X cold boot editor, text is cleared <p>In save menu only</p> <ul style="list-style-type: none"> • ctrl-V save to floppy with file type Assembler
7153	ED warm	

ROM Menu

7000	RM	<p>For easier access to the ROM utilities there is a menu program in the PRT-MPF spare ROM at 7000.</p> <p>Operation</p> <ul style="list-style-type: none"> • Start with <G>7000 or use CTRL- B from the monitor • The menu is displayed, switch with SPACE between the two lines • Select the desired function • Terminate with SHI FT- CONTROL <p>Menu I tems</p> <ul style="list-style-type: none"> • <u>R</u>- RB Receive Binary * • CTRL- <u>R</u> Receive Binary for <u>A</u>- ASM/ED or <u>B</u>- BASI C ** Editor or Basic must be initialized before data can be received. If it is not, respond with Y to the INITIALI ZE ? prompt, the editor or Basic is then cold booted. N continues receiving. • <u>S</u>- SB Send Binary * • CTRL- <u>S</u> Send Binary for <u>A</u>- ASM/ED or <u>B</u>- BASI C ** Editor or Basic must be initialized before data can be sent. If it is not, respond with Y to the INITIALI ZE ? prompt, the editor or Basic is then cold booted. N continues sending. • <u>H</u>- RH Receive HEX * • <u>V</u>- VC VC-1541 floppy menu * • <u>W</u>- SW Stopwatch * • <u>E</u>- ED Text Editor, tries to warm boot • <u>B</u>- BA Basic cold start <p>* Shown if IOM board connected ** Basic is cold-booted if not started since power-up</p>
------	----	---

Hardware

Memory Map

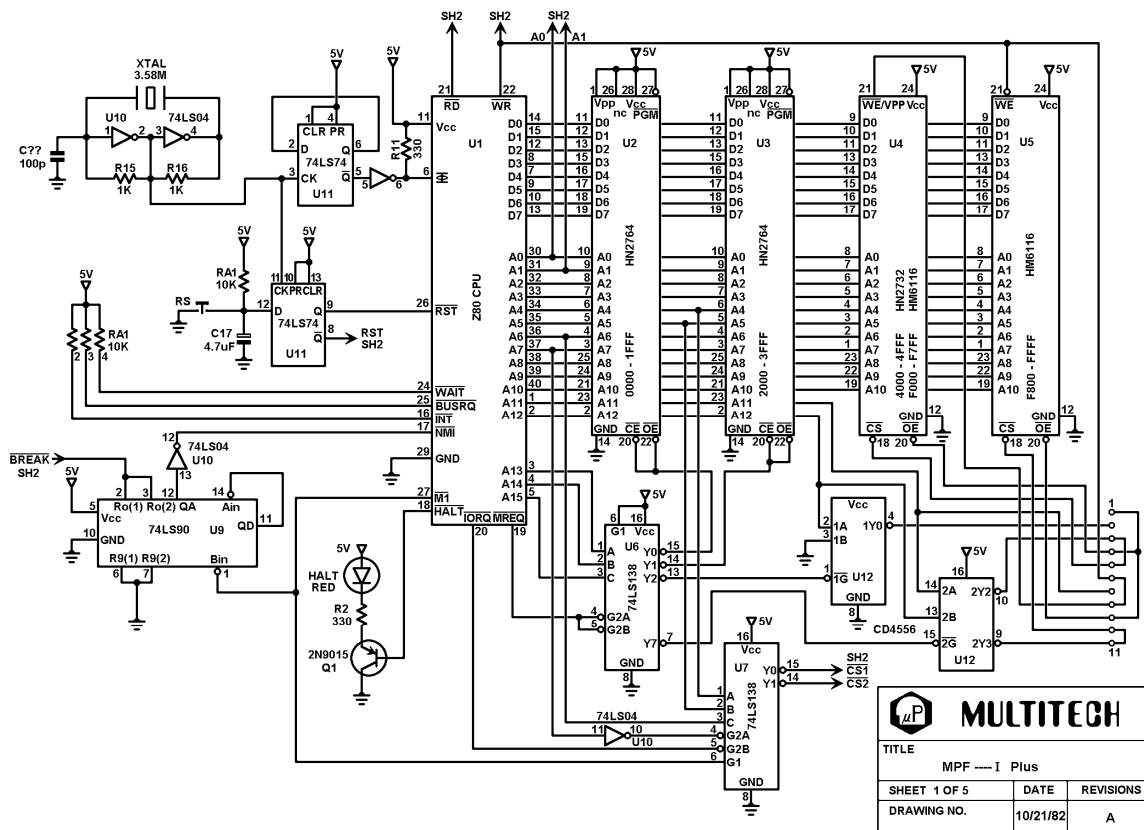
0000-1FFF	MPF U2 Monitor/Assembler
2000-3FFF	MPF U3 Basic/Forth
6000-6FFF	PRT-MPF U5 Printer firmware
7000-7FFF	PRT-MPF U6 Spare ROM (RomMenu, Editor, Library)
B000-BFFF	IOM-MPF ROM (RcvHex, RcvBin, SendBin, StopWatch, Library)
D800-DFFF	VC-1541 Floppy driver, replaces original 6116 RAM
E000-EFFF	IOM-MPF RAM
F000-FFFF	MPF RAM

I/O Ports

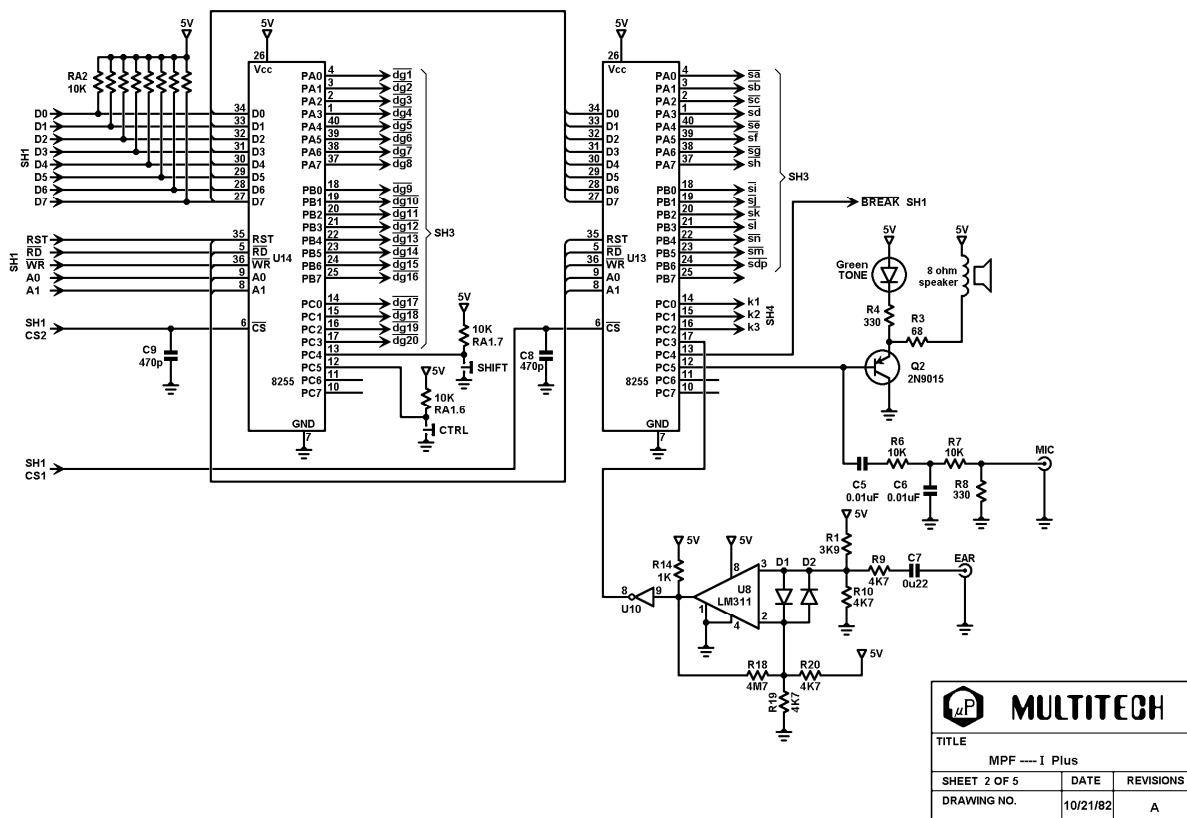
60H	dpIUD	MPF-IOM 8251 USART, Data
61H	dpIUC	MPF-IOM 8251 USART, Control
64H	dpIC0	MPF-IOM Z80-CTC, Channel 0
65H	dpIC1	MPF-IOM Z80-CTC, Channel 1
66H	dpIC2	MPF-IOM Z80-CTC, Channel 2 (used as baudrate generator for 8251)
67H	dpIC3	MPF-IOM Z80-CTC, Channel 3
68H	dpIPAD	MPF-IOM Z80-PIO, Channel A Data
69H	dpIPBD	MPF-IOM Z80-PIO, Channel B Data
6AH	dpIPAC	MPF-IOM Z80-PIO, Channel A Control
6BH	dpIPBC	MPF-IOM Z80-PIO, Channel B Control
6CH	dpOSW	MPF-IOM DIP-Switch
80H	dpMDD1	MPF 8255 PPI1, Port A (display digits 1-8)
81H	dpMDD2	MPF 8255 PPI1, Port B (display digits 9-16)
82H	dpMDD3	MPF 8255 PPI1, Port C (display digits 17-20, SHIFT & CONTROL keys)
90H	dpMDS1	MPF 8255 PPI2, Port A (display segments a to h)
91H	dpMDS2	MPF 8255 PPI2, Port B (display segments i to dp)
92H	dpMKT	MPF 8255 PPI2, Port C (tape in-out/keyboard inputs)

MPF-1P Schematics

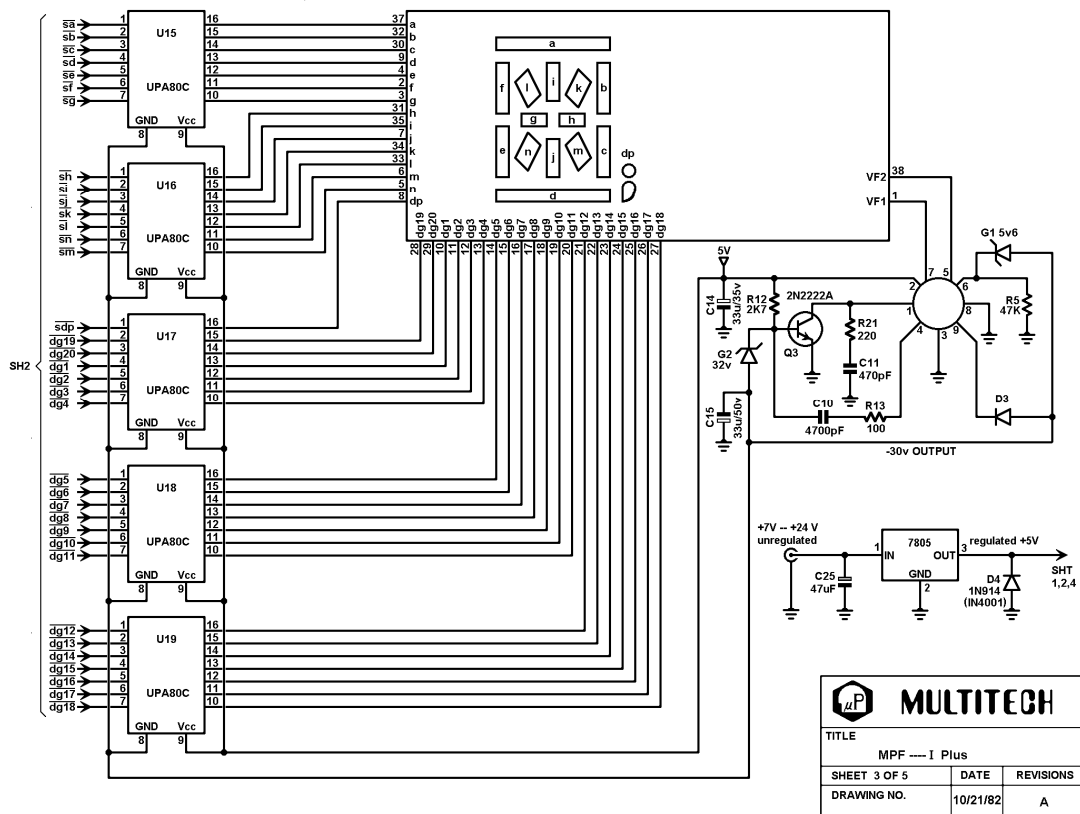
Clock, CPU, Memory



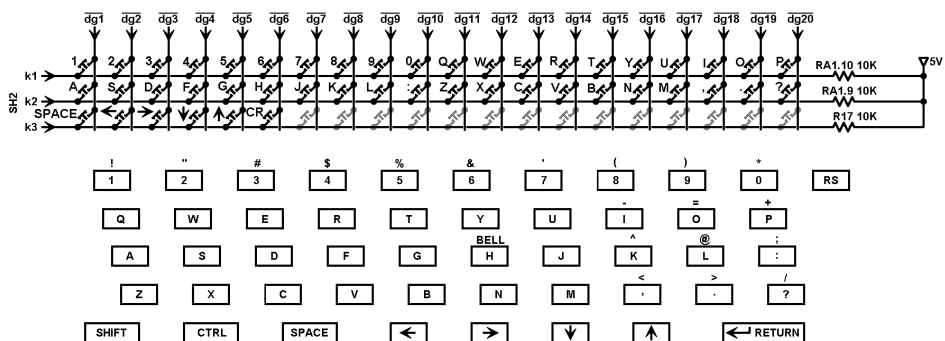
PPI, Speaker, Tape I/O



Display



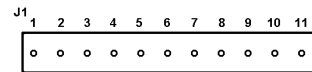
Keyboard Matrix



Power Supply, Connectors

J1 PIN FUNCTION			
PIN NO.	SIGNAL	PIN NO.	SIGNAL
1	A11	21	A10
2	A12	22	A9
3	A13	23	A8
4	A14	24	A7
5	A15	25	A6
6	CK	26	A5
7	D4	27	A4
8	D3	28	A3
9	D5	29	A2
10	D6	30	A1
11	NC	31	A0
12	D2	32	GND
13	D7	33	RFSH
14	D0	34	M1
15	D1	35	RESET
16	INT	36	BUSRQ
17	NMI	37	WAIT
18	HALT	38	BUSAK
19	MREQ	39	WR
20	IORQ	40	RD

J2 PIN FUNCTION	
PIN NO.	SIGNAL
1	Q5 - C
2	U12 - 4
3	U12 - 10
4	U4 - 20
5	U4 - 18
6	WR
7	A11
8	U4 - 21
9	U5 - 20
10	U5 - 18
11	U12 - 9



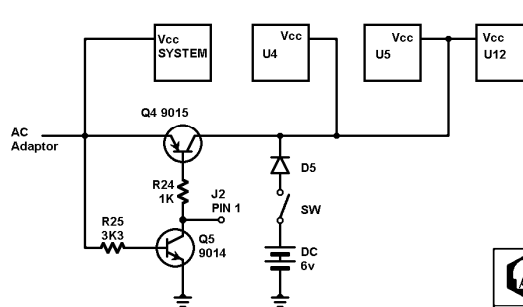
U4 U5 Default connection is designed for 6116

J2
PIN 1,4,9 Short
PIN 3,5 Short
PIN 6,8 Short
PIN 10,11 Short

If users want to change 6116 into 5516 connection for lower power battery backup

first cut
PIN 1,4,9
PIN 3,5
PIN 10,11

second connect
PIN 1,5,10
PIN 3,4
PIN 9,11

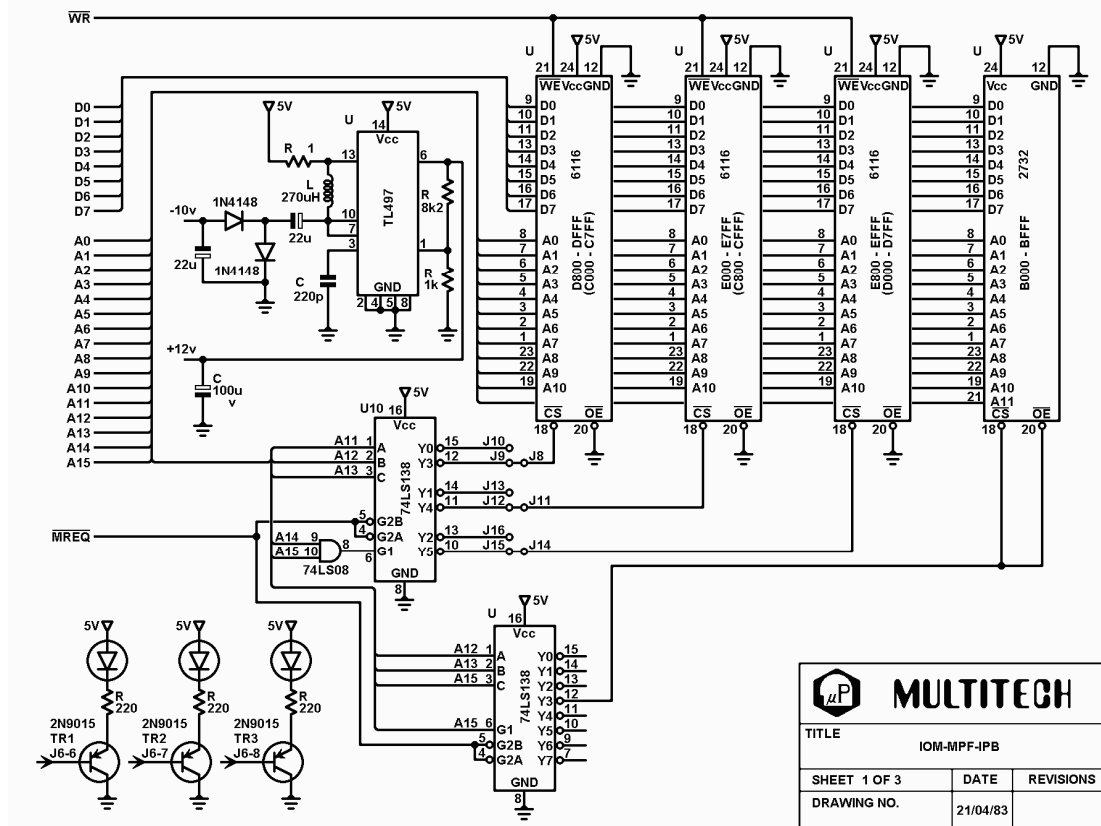


SW	Description
ON	auto battery back up
OFF	No battery backup

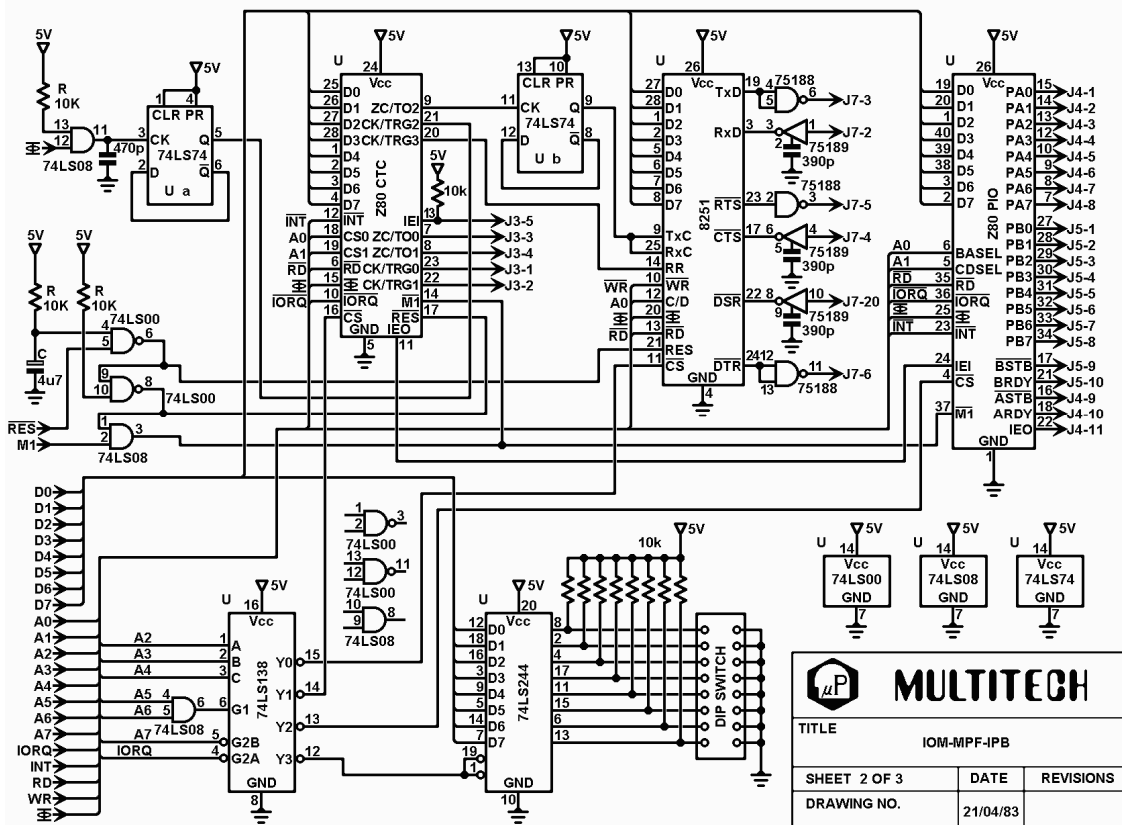
TITLE		
MPF -- I Plus		
SHEET 5 OF 5	DATE	REVISIONS
DRAWING NO.	10/21/82	A

IOM-MPF Schematics

RAM ROM LEDs, Address Decoding



PIO, CTC, DIP Switch



Connectors

J1		J2	
1	A11	21	A10
2	A12	22	A9
3	A13	23	A8
4	A14	24	A7
5	A15	25	A6
6	CK	26	A5
7	D4	27	A4
8	D3	28	A3
9	D5	29	A2
10	D6	30	A1
11	NC	31	A0
12	D2	32	GND
13	D7	33	RFSH
14	D0	34	M1
15	D1	35	RESET
16	INT	36	BUSRQ
17	NMI	37	WAIT
18	HALT	38	BUSAK
19	MREQ	39	WR
20	IORQ	40	RD


J6	
1	INT
2	NMI
3	HALT
4	MREQ
5	IORQ
6	M1
7	RESET
8	WAIT
9	BUSRQ
10	BUSAK
11	WR
12	RD
13	CK
14	NC
15	NC
16	NC

J5	
1	PA0
2	PA1
3	PA2
4	PA3
5	PA4
6	PA5
7	PA6
8	PA7
9	ASTB
10	ARDY
11	PIOEIO
12	NC
13	NC
14	NC
15	NC
16	NC

J4	
1	PB0
2	PB1
3	PB2
4	PB3
5	PB4
6	PB5
7	PB6
8	PB7
9	BSTB
10	BRDY
11	NC
12	NC
13	NC
14	NC
15	NC
16	NC

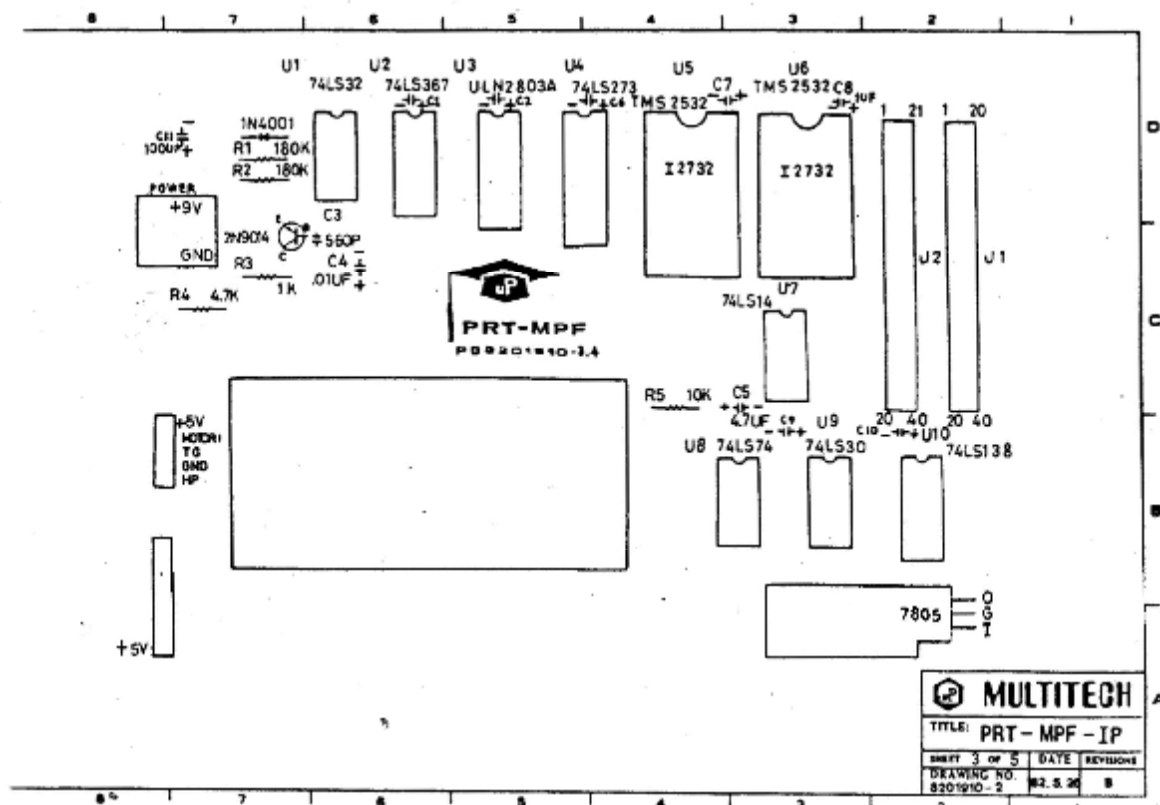
J3	
1	CK/TR0
2	CK/TR1
3	ZC/TO0
4	ZC/TO1
5	CTCIEI
6	TR1
7	TR2
8	TR3
9	+5V
10	+5V
11	NC
12	GND
13	GND
14	NC
15	-10V
16	+12V

J7 or J17	
2	RxD
3	TxD
4	CTS
5	RTS
6	DTR
7	GND
20	DSR

 MULTITECH			
		TITLE	
		IOM-MPF-IPB	
SHEET 3 OF 3	DATE	REVISIONS	
DRAWING NO.	21/04/83		

PRT-MPF Schematics

Board Layout



Schematics

