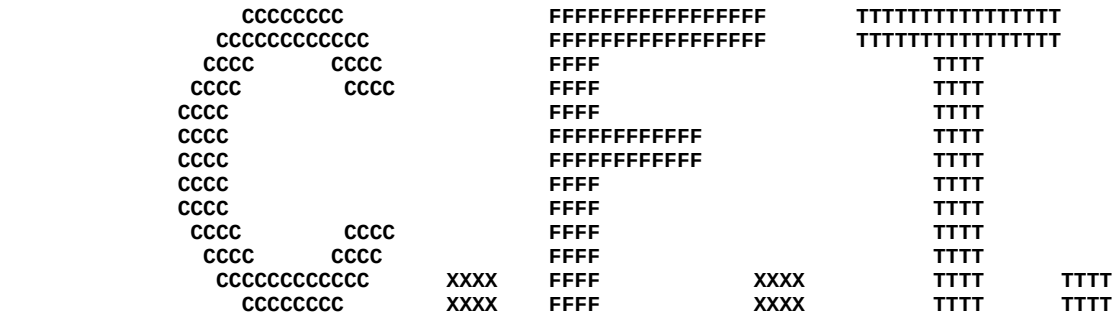

R E F E R E N C E C A R D

A 16-bit mini computer in the 1970s sense, designed and built out of 74xxx integrated circuits, and heavily influenced by the design of the DEC PDP-8. Everything from the instruction set to the operating system and documentation is home-brew. This is a bus and instruction set cheat sheet. The instruction set of the CFT cannot be quantified easily due to the PDP-8 styled instructions and extensions, but the 76 most commonly used instructions are tabulated here.

CONTROL BUS (P1)

1 CLL#	2 CPL#	3 RAGL#	4 WALU#
5 FL	6 FV	7 IR0	8 WEN#
9 IR2	10 END#	11 RUNIT0	12 RUNIT1
13 RUNIT2	14 RUNIT3	15 SKIP#	16 STI#
17 CLI#	18 OPIF0	19 OPIF1	20 OPIF2
21 OPIF3	22 IR11	23 AC0	24 AC1
25 AC2	26 AC3	27 AC4	28 AC5
29 AC6	30 AC7	31 AC8	32 AC9
33 AC10	34 AC11	35 AC12	36 AC13
37 AC14	38 AC15	39 ACCPL#	40 DEC#

CONTROL BUS (P2)

1 SPTAC#	2 STPDR#	3 INCPC#	4 RAC#
5 RDR#	6 RPC#	7 WAC#	8 WAR#
9 WDR#	10 WPC#	11 FNEG	12 FZERO
13 PC10	14 PC11	15 PC12	16 PC13
17 PC14	18 PC15	19 WIR#	20 AINDEX#
21 IBUS0	22 IBUS1	23 IBUS2	24 IBUS3
25 IBUS4	26 IBUS5	27 IBUS6	28 IBUS7
29 IBUS8	30 IBUS9	31 IBUS10	32 IBUS11
33 IBUS12	34 IBUS13	35 IBUS14	36 IBUS15
37 IR12	38 IR13	39 IR14	40 IR15

EXPANSION BUS (DIN 41612)

A1 GND	B1 GND	C1 GND	
A2 GND	B2 GND	C2 GND	
A3 +5V	B3 +5V	C3 +5V	
A4 +3.3V	B4 +3.3V	C4 +3.3V	
A5 AB0 >	B5 NC x	C5 DB0 =	
A6 AB1 >	B6 SKIPEXT# *	C6 DB1 =	
A7 AB2 >	B7 ENDEXT# *	C7 DB2 =	
A8 AB3 >	B8 WS# *	C8 DB3 =	
A9 AB4 >	B9 R1# >	C9 DB4 =	
A10 AB5 >	B10 R6# >	C10 DB5 =	
A11 AB6 >	B11 R7# >	C11 DB6 =	
A12 AB7 >	B12 W1# >	C12 DB7 =	
A13 IRQ3# *	B13 AEXT0 >	C13 MEM# >	
A14 IRQ4# *	B14 AEXT1 >	C14 IO# >	
A15 IRQ5# *	B15 AEXT2 >	C15 R# >	
A16 IRQ6# *	B16 AEXT3 >	C16 W# >	
A17 TPA -	B17 AEXT4 >	C17 TPC -	
A18 IRQ7#	B18 AEXT5 >	C18 IRQ0# *	
A19 HALT# *	B19 AEXT6 >	C19 IRQ1# *	
A20 AB8 >	B20 AEXT7 >	C20 IRQ2# *	
A21 AB9 >	B21 NC x	C21 DB8 =	
A22 AB10 >	B22 RSTHOLD# >	C22 DB9 =	
A23 AB11 >	B23 NC x	C23 DB10 =	
A24 AB12 >	B24 IODEV1XX# >	C24 DB11 =	
A25 AB13 >	B25 IODEV2XX# >	C25 DB12 =	
A26 AB14 >	B26 IODEV3XX# >	C26 DB13 =	
A27 AB15 >	B27 T34# >	C27 DB14 =	
A28 SYSDEV#	B28 CLK4 >	C28 DB15 =	
A29 CLK1 >	B29 CLK2 >	C29 IRQ# *	
A30 IRQS# >	B30 CLK3 >	C30 RESET# *	
A31 GND	B31 GND	C31 GND	
A32 +5V	B32 +5V	C32 +5V	

Active low
> Output
< Input
***** Open Drain
= Bidirectional bus
x Do not connect
- Card-local signal

http://www.bedroomlan.org/cft

Mnemonic	Hex	Op..IR...Operand	Cycles	INZVL	MIXC	TOme	Description. Semantics. Notes.
TRAP	0000	00000Raaaaaaaaaa	5	-----	A---	F---	Save PC and jump to trap. [1]=PC; PC=a
TRAP I	0800	00001Rmmmmmmmmmm	7/9	-----	AIX-	F---	Save PC and jump to trap ind. [1]=PC; PC=[m]
IOT	1000	00010Rddddddddddd	5	-NZ--	D---	I---	I/O transaction. {d}=AC; AC={d}
IOT I	1800	00011Rmmmmmmmmmm	7/9	-NZ--	DIX-	I---	I/O transaction ind. {[m]}=AC; AC={[m]+}
LOAD	2000	00100Rddddddddddd	5	-NZ--	D---	M---	Load from memory. AC=[m]
LOAD I	2800	00101Rmmmmmmmmmm	7/9	-NZ--	DIX-	M---	Load from memory, ind. AC=[[m]+]
STORE	3000	00110Rddddddddddd	5	-----	D---	M---	Store to memory. AC=[m]
STORE I	3800	00111Rmmmmmmmmmm	7/9	-----	DIX-	M---	Store to memory, ind. AC=[[m]+]
PUSH	3C00	001111001mmmmmmm	9	-----	--X-	M---	Push onto stack pointer m. Macro: STORE I R m. **1,2
IN	4000	01000Rddddddddddd	4	-NZ--	D---	I---	Input from device. AC={d}
IN I	4800	01001Rmmmmmmmmmm	6/8	-NZ--	DIX-	I---	Input from device, ind. AC={[m]+}
OUT	5000	01010Rddddddddddd	4	-----	D---	I---	Output to device. {d}=AC
OUT I	5800	01011Rmmmmmmmmmm	6/8	-----	DIX-	I---	Output to device, ind. {[m]+}=AC
JMP	6000	01100Raaaaaaaaaa	3	-----	A---	F---	Jump. PC=a
JMP I	6800	01101Rmmmmmmmmmm	4/7	-----	AIX-	F---	Jump, ind. PC=[a]+
RET	6C00	0110110000000000	4	-----	a---	F-m-	Return from subroutine. M: JMP I R 0. **3
RTT	6C01	0110110000000001	4	-----	a---	F-m-	Return from trap. M: JMP I R 1. **3
RTI	6C02	0110110000000010	4	-----	a---	F-m-	Return from interrupt. M: JMP I R 2. **3
JSR	7000	01110Raaaaaaaaaa	5	-----	A---	F---	Save PC and jump to subroutine. [0]=PC; PC=a
JSR I	7800	01111Rmmmmmmmmmm	6/9	-----	AIX-	F---	Save PC and jump to subroutine, ind. [0]=PC; PC=[a]+
ADD	8000	10000Rmmmmmmmmmm	5	-NZVL	D---	A---	Add memory to AC. <L,AC>=<L,AC> + [a]
ADD I	8800	10001Rmmmmmmmmmm	7/9	-NZVL	DIX-	A---	Add memory to AC, inc. <L,AC>=<L,AC> + [[m]+]
AND	9000	10010Rmmmmmmmmmm	5	-NZ--	D---	A---	Bitwise AND memory and AC. AC=AC & [m]
AND I	9800	10011Rmmmmmmmmmm	7/9	-NZ--	DIX-	A---	Bitwise AND memory and AC, inc. AC=AC & [[m]+]
OR	A000	10100Rmmmmmmmmmm	5	-NZ--	D---	A---	Bitwise OR memory and AC. AC=AC | [m]
OR I	A800	10101Rmmmmmmmmmm	7/9	-NZ--	DIX-	A---	Bitwise OR memory and AC, inc. AC=AC | [[m]+]
XOR	A000	10100Rmmmmmmmmmm	5	-NZ--	D---	A---	Bitwise XOR memory and AC. AC=AC ^ [m]
XOR I	A800	10101Rmmmmmmmmmm	7/9	-NZ--	DIX-	A---	Bitwise XOR memory and AC, inc. AC=AC ^ [[m]+]
OP1	C000	11000xbbbbbbbbbb	10	-??-?	---C	?1--	Combined ops 1. **4
NOP10	C000	11000x0000000000	10	-----	---C	-1--	No operation, ten idle cycles. M: OP1. **3
IFL	C200	11000x1---------	3>10	-----	---C	F1--	Execute remaining instr. iff L set. !L => end.
IFV	C100	11000x-1--------	4>10	-----	---C	F1--	Execute remaining instr. iff V set. !V => end.
CLA	C080	11000x--1-------	10	-00--	---C	-1--	Clear AC. AC=0
CLL	C040	11000x---1------	10	----0	---C	-1--	Clear L. L=0
NOT	C020	11000x----1-----	10	-NZ--	---C	A1--	Complement AC. AC=!AC
INC	C010	11000x-----1----	10	-NZ-L	---C	A1--	Increment. <L,AC>++
CPL	C008	11000x------1---	10	----*	---C	-1--	Complement link. L=!L
RBL	C002	11000x-------010	10	-NZ-L	---C	A1--	Roll bit left. <L,AC>=<L,AC> <- 1. **5
RBR	C001	11000x-------001	10	-NZ-L	---C	A1--	Roll bit right. <L,AC>=<L,AC> -> 1. **5
RNL	C00A	11000x-------110	10	-NZ-L	---C	A1--	Roll nybble left <L,AC>=<L,AC> <- 4. **5
RNR	C005	11000x-------101	10	-NZ-L	---C	A1--	Roll nybble right <L,AC>=<L,AC> -> 4. **5
NEG	C030	11000x----11----	10	-NZ--	---C	A1m-	Two’s complement. M: NOT INC.
ING	C020	11000x----1-----	10	-NZ--	---C	A1m-	INcrement and neGate (One’s complement). M: NOT.
SBL	C042	11000x---1---010	10	-NZ-L	---C	A1m-	Shift bit left without sign extension. M: CLL RBL.
SBR	C041	11000x---1---001	10	-NZ-L	---C	A1m-	Shift bit right without sign extension. M: CLL RBR.
SEL	C048	11000x---1--1---	10	----L	---C	-1m-	Set L. M: CLL CPL
OP2	D000	11010xbbbbbbbbbb	8	???--	---C	?2--	Combined ops 2. **4
NOP8	D000	11010x0000000000	8	-----	---C	-2--	No operation, eight idle cycles. **3
SKPN	D000	11010x-----00000	8	-----	---C	F2--	Skip never. **6
SNA	D008	11010x-----01---	8	-----	---C	F2--	Skip if negative. N => PC++. **6
SZA	D004	11010x-----0-1--	8	-----	---C	F2--	Skip if zero. Z => PC++. **6
SSL	D002	11010x-----0--1-	8	-----	---C	F2--	Skip if link. L => PC++. **6
SSV	D001	11010x-----0---1	8	-----	---C	F2--	Skip if overflow. V => PC++. **6
SNP	D00C	11010x-----01100	8	-----	---C	F2m-	Skip if non-positive. M: SNA SZA. **6
Mnemonic	Hex	Op..IR...Operand	Cycles	INZVL	MIXC	TOme	Description. Semantics. Notes.

Mnemonic	Hex	Op..IR...Operand	Cycles	INZVL	MIXC	TOme	Description. Semantics. Notes.
SKIP	D010	11010x-----10000	8	-----	---C	F2--	Skip always. PC=PC+1. **7
SNN	D018	11010x-----11---	8	-----	---C	F2--	Skip if non-negative. !N => PC++. **7
SNZ	D014	11010x-----1-1--	8	-----	---C	F2--	Skip if non-zero. !N => PC++. **7
SCL	D012	11010x-----1--1-	8	-----	---C	F2--	Skip if link clear. !L => PC++. **7
SCV	D011	11010x-----1---1	8	-----	---C	F2--	Skip if no overflow. !V => PC++. **7
SPA	D01C	11010x-----11100	8	-----	---C	F2m-	Skip if positive. M: SNN SNZ. **7
CLA	D080	11010x--1-------	8	-00--	---C	-2--	Clear AC. AC=0
CLI	D040	11010x---1 -----	8	0----	---C	-2--	Clear Interrupts. I=0.
STI	D020	11010x----1-----	8	1----	---C	-2--	Set Interrupts. I=1.
POP	D000	11011Rmmmmmmmmmm	8	-NZ--	--X-	M2--	Pop from stack (decrement and load). [m]--; AC=[[m]]. **2
ISZ	E000	11100Rmmmmmmmmmm	7	-NZ-L	D---	F---	Increment and skip if zero. AC=[m]+1; [m]=AC; Z => PC+
ISZ I	E800	11101Rmmmmmmmmmm	9/11	-NZ-L	DIX-	F---	Increment ind. & skip if 0. AC=[[m]]+1; [[m]+]=AC; Z => PC++
LIA	F000	11110Raaaaaaaaaa	3	-NZ--	A---	----	Load address/literal. AC=a
LI	F400	111101iiiiiiiiii	3	-NZ--	L---	--m-	Load immediate. M: LIA R
JMPII	F800	11111Rmmmmmmmmmm	6/8	-----	A2X-	F---	Jump doubly indirect (Forth ENTER). PC = [[m]+]
SBP	5400	010101000000nnnn	4	-----	D---	I-me	Select Microcode Bank n. {UBCp}=n. M: OUT R &n. **1,8
SBN	5410	010101000001nnnn	4	-----	D---	I-me	Set uC Bank for next instr. {UBCt}=n. M: OUT R &1n. **1,8,B
SMB	5420	0101010000100nnn	4	-----	D---	I-me	Set Memory Bank n. {MBRn}=AC. M: OUT R &2n. **1,9
SOR	5432	0101010000110010	4	-----	d---	I-me	Set Front Panel Output Register. {OR}=AC. M: OUT R &32. **A
HALT	5437	0101010000110111	4	-----	d---	I-me	Halt clock. M: OUT R &37. **1,A
LSR	4430	0100010000110000	4	-NZ--	d---	I-me	Read Front Panel Switch Regstr. AC={SR}. M: IN R &30. **1,A
LDSR	4431	0100010000110001	4	-NZ--	d---	I-me	Read Front Panel DIP Switches. AC={DSR}. M: IN R &31. **1,A

Mnemonic	Hex	Op..IR...Operand	Cycles	INZVL	MIXC	TOme	Description. Semantics. Notes.
		1					Bit must be set
		0					Bit must be clear
		-					Any value will do (don’t care / OR with other bit values)
		Op..					Opcode
		I					Indirection bit
		Op..I					Instruction microprogram identifier
		R					Register bit. If R=1: zero page. R=0: relative to PC.
		aaaaaaaaaa					Address operand (address modes)
		mmmmmmmmmm					Address operand (direct modes)
		bbbbbbbbbb					Bitmap operand (minor operations)
		iiiiiiiiii					Immediate operand (LI instruction)
		nnnnnnnnnn					Unit number (extensions)
			nn				Runs in n clock cycles
			/mm				Autoindex runs in m clock cycles
			nn>cc				Runs in n cycles if false, c if true
				?			Flag may be affected depending on operand
				0			Flag cleared
				1			Flag set
				*			Flag toggled
				-			Flag unaffected
				I			Interrupt flag affected
				N			Negative flag affected
				Z			Zero flag affected
				V			Overflow flag affected
				L			Link affected

| Mnemonic | Hex | Op..IR...Operand | Cycles | INZVL | MIXC | TOme | Description. Semantics. Notes. |

Mnemonic	Hex	Op..IR...Operand	Cycles	INZVL	MIXC	TOme	Description. Semantics. Notes.
					A		(Register) Address mode
					D		(Register) Direct mode
					L		Literal mode
					a		Implied address
					d		Implied direct mode address
					I		Indirect
					2		Double Indirect
					X		Autoindex if &080 <= m <= &FF.
					C		Conditional
						F	Flow control
						M	Memory space
						I	I/O space
						A	Arithmetic/Logic
						?	Depends on operand
						1	OP1 Minor Operations
						2	OP2 Minor Operations
						m	Instruction Macro/Alias
						e	Extension
							AC Accumulator
							L Link/Carry register
							<L,AC> 17-bit aggregate, L is most-significant bit.
							+ Bitwise addition.
							X++ Shorthand for X = X + 1
							X-- Shorthand for X = X - 1
							PC Program Counter
							[n] memory at address n
							[[n]] indirection: memory at address [n]
							[n]+ autoindex: if &80<=n<=&FF, [n]=[n]+1 at end of instr.
							{n} I/O space at address n
							X=Y X gets the value of Y
							!X => ACTION Perform ACTION if flag X is clear.
							end End execution of current instruction.
							& Bitwise And
							| Bitwise Or
							^ Bitwise Exclusive Or (1 if bits differ)
							!X Bitwise Not of X (toggle bit(s))
							X <- n Roll bits in X n positions to the left.
							X -> n Roll bits in X n positions to the right.
							M: Macro (actual instruction executed follows)
							**1 Not all 10 bits of operand are used/allowed.
							**2 Works only for autoindex addresses (&80-&FF)
							**3 No operand.
							**4 Combined minor operations. Executed top-to-bottom as listed. Operand fragments can be ORred together to combine multiple minor operations.
							**5 Roll operation. L acts as 17th bit. Roll operations are mutually exclusive.
							**6 Group 1 conditionals. Mutually exclusive with group 2.
							**7 Group 2 conditionals. Mutually exclusive with group 1.
							**8 Microcode Banking Extension (UCB) must be present.
							**9 Memory Banking Unit (MBU) must be present.
							**A Programmer’s Front Panel (PFP) must be present.
							**B May delay intrpt. handling by 1 fetch-execute cycle.