Texas Instruments

TMS320C80
Digital Signal Processor

Data Sheet

1997 Digital Signal Processing Solutions

TEXAS INSTRUMENTS

Printed in U.S.A., October 1997

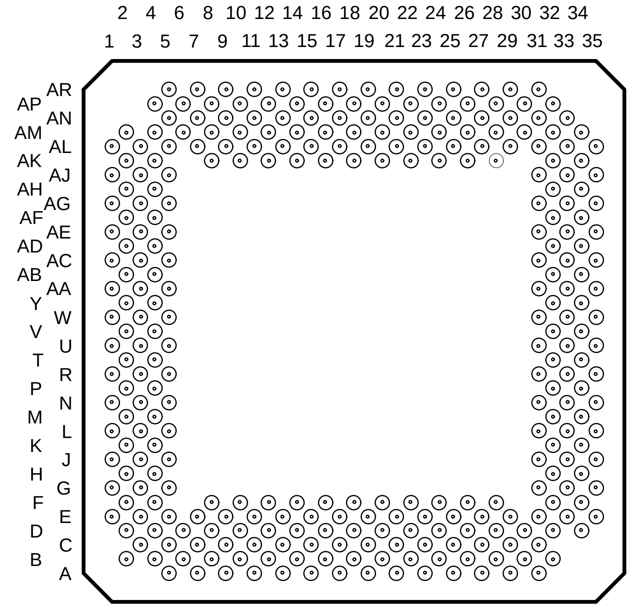
Data Sheet

TMS320C80 DSP

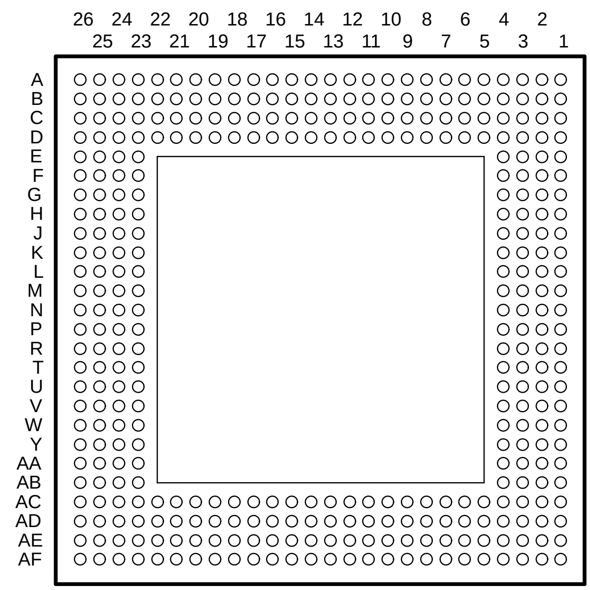
1997

- **Single-Chip Parallel Multiple Instruction/Multiple Data (MIMD) DSP**
- **More Than Two Billion RISC-Equivalent Operations per Second**
- **Master Processor (MP)**
  - **32-Bit Reduced Instruction Set Computing (RISC) Processor**
  - **IEEE-754 Floating-Point Capability**
  - **4K-Byte Instruction Cache**
  - **4K-Byte Data Cache**
- **Four Parallel Processors (PP)**
  - **32-Bit Advanced DSPs**
  - **64-Bit Opcode Provides Many Parallel Operations per Cycle**
  - **2K-Byte Instruction Cache and 8K-Byte Data RAM per PP**
- **Transfer Controller (TC)**
  - **64-Bit Data Transfers**
  - **Up to 480M-Byte/s Transfer Rate**
  - **32-Bit Addressing**
  - **Direct DRAM/VRAM Interface With Dynamic Bus Sizing**
  - **Intelligent Queuing and Cycle Prioritization**
- **Video Controller (VC)**
  - **Provides Video Timing and VRAM Control**
  - **Dual-Frame Timers for Two Simultaneous Image-Capture and/or Display Systems**
- **Big- or Little-Endian Operation**
- **50K-Byte On-Chip RAM**
- **4G-Byte Address Space**
- **16.6-ns Cycle Time**
- **3.3-V Operation**
- **IEEE Standard 1149.1† Test Access Port (JTAG)**

GF PACKAGE
(BOTTOM VIEW)



GGP PACKAGE
(BOTTOM VIEW)



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

† IEEE Standard 1149.1–1990, IEEE Standard Test Access Port and Boundary-Scan Architecture

description

The TMS320C80 is a single chip, MIMD parallel processor capable of performing over two billion operations per second. It consists of a 32-bit RISC master processor with a 120-MFLOP IEEE floating-point unit, four 32-bit parallel processing digital signal processors (DSPs), a transfer controller with up to 480M-byte/s off-chip transfer rate, and a video controller. All the processors are coupled tightly through an on-chip crossbar that provides shared access to on-chip RAM. This performance and programmability make the ’C80 ideally suited for video, imaging, and high-speed telecommunications applications.

GF Terminal Assignments – Numerical Listing

TERMINAL		TERMINAL		TERMINAL		TERMINAL	
NO.	NAME	NO.	NAME	NO.	NAME	NO.	NAME
A5	CT1	C21	V_{DD}	E33	HSYNC0	L5	V_{SS}
A7	V_{DD}	C23	W	E35	TCK	L31	V_{SS}
A9	HACK	C25	DBEN	F2	V_{DD}	L33	TRST
A11	V_{SS}	C27	V_{SS}	F4	V_{SS}	L35	XPT1
A13	CAS/DQM7	C29	CAREA0	F8	V_{DD}	M2	V_{DD}
A15	CAS/DQM5	C31	CBLNK0 / VBLNK0	F10	V_{SS}	M4	V_{SS}
A17	V_{DD}	D2	RETRY	F12	V_{DD}	M32	V_{SS}
A19	V_{SS}	D4	V_{DD}	F14	PS0	M34	V_{DD}
A21	RAS	D6	V_{SS}	F16	V_{SS}	N1	V_{DD}
A23	DSF	D8	AS0	F18	CT2	N3	A8
A25	V_{SS}	D10	UTIME	F20	V_{DD}	N5	V_{SS}
A27	SCLK1	D12	V_{SS}	F22	V_{SS}	N31	V_{SS}
A29	V_{DD}	D14	RESET	F24	V_{DD}	N33	TMS
A31	EINT1	D16	REQ0	F26	V_{SS}	N35	V_{DD}
B2	No Connect	D18	V_{SS}	F28	V_{DD}	P2	A4
B4	BS1	D20	CAS/DQM0	F32	V_{SS}	P4	A9
B6	V_{DD}	D22	FCLK1	F34	V_{DD}	P32	TDO
B8	PS1	D24	V_{SS}	G1	V_{DD}	P34	XPT0
B10	REQ1	D26	CAREA1	G3	A2	R1	V_{SS}
B12	V_{DD}	D28	SCLK0	G5	A1	R3	V_{DD}
B14	CAS/DQM6	D30	V_{SS}	G31	EINT2	R5	V_{DD}
B16	CAS/DQM3	D32	V_{DD}	G33	CBLNK1 / VBLNK1	R31	V_{DD}
B18	V_{DD}	D34	VSYNC0	G35	V_{DD}	R33	V_{DD}
B20	CAS/DQM1	E1	AS1	H2	STATUS0	R35	V_{SS}
B22	TRG/CAS	E3	FAULT	H4	A3	T2	A5
B24	V_{DD}	E5	V_{SS}	H32	CSYNC1 / HBLNK1	T4	A13
B26	DDIN	E7	STATUS2	H34	TDI	T32	D62
B28	FCLK0	E9	READY	J1	STATUS1	T34	EMU0
B30	V_{DD}	E11	BS0	J3	V_{SS}	U1	V_{DD}
B32	CSYNC0 / HBLNK0	E13	V_{SS}	J5	V_{DD}	U3	A10
C3	V_{SS}	E15	HREQ	J31	V_{DD}	U5	PS3
C5	STATUS3	E17	CAS/DQM4	J33	V_{SS}	U31	FF1
C7	AS2	E19	RL	J35	EMU1	U33	D61
C9	V_{SS}	E21	STATUS5	K2	STATUS4	U35	V_{DD}
C11	CT0	E23	V_{SS}	K4	A6	V2	V_{DD}
C13	PS2	E25	CLKOUT	K32	VSYNC1	V4	V_{SS}
C15	V_{DD}	E27	LINT4	K34	HSYNC1	V32	V_{SS}
C17	CLKIN	E29	EINT3	L1	A0	V34	V_{DD}
C19	CAS/DQM2	E31	V_{SS}	L3	A7	W1	A11

GF Terminal Assignments – Numerical Listing (Continued)

TERMINAL		TERMINAL		TERMINAL		TERMINAL	
NO.	NAME	NO.	NAME	NO.	NAME	NO.	NAME
W3	A18	AG1	A16	AL17	D20	AN29	D35
W5	V_{SS}	AG3	V_{SS}	AL19	D21	AN31	D45
W31	V_{SS}	AG5	V_{DD}	AL21	D24	AN33	V_{DD}
W33	D59	AG31	V_{DD}	AL23	V_{SS}	AP4	A27
W35	D63	AG33	V_{SS}	AL25	D29	AP6	V_{DD}
Y2	A12	AG35	D57	AL27	D32	AP8	D5
Y4	A19	AH2	A20	AL29	D38	AP10	D8
Y32	$\overline{XPT2}$	AH4	A30	AL31	V_{SS}	AP12	V_{DD}
Y34	D56	AH32	D44	AL33	D48	AP14	D13
AA1	V_{SS}	AH34	D54	AL35	D53	AP16	D17
AA3	V_{DD}	AJ1	V_{DD}	AM2	A24	AP18	V_{DD}
AA5	V_{DD}	AJ3	A31	AM4	V_{DD}	AP20	D26
AA31	V_{DD}	AJ5	V_{SS}	AM6	V_{SS}	AP22	D34
AA33	V_{DD}	AJ31	V_{SS}	AM8	D2	AP24	V_{DD}
AA35	V_{SS}	AJ33	D42	AM10	D6	AP26	D39
AB2	A14	AJ35	V_{DD}	AM12	V_{SS}	AP28	D41
AB4	A21	AK2	V_{DD}	AM14	D14	AP30	V_{DD}
AB32	D55	AK4	V_{SS}	AM16	D19	AP32	D47
AB34	D60	AK8	V_{DD}	AM18	V_{SS}	AR5	D0
AC1	V_{DD}	AK10	V_{SS}	AM20	D23	AR7	V_{DD}
AC3	A22	AK12	V_{DD}	AM22	D25	AR9	D7
AC5	V_{SS}	AK14	V_{SS}	AM24	V_{SS}	AR11	V_{SS}
AC31	V_{SS}	AK16	V_{DD}	AM26	D31	AR13	D11
AC33	D52	AK18	FF2	AM28	D33	AR15	D15
AC35	V_{DD}	AK20	V_{SS}	AM30	V_{SS}	AR17	V_{SS}
AD2	V_{DD}	AK22	D27	AM32	V_{DD}	AR19	V_{DD}
AD4	V_{SS}	AK24	V_{DD}	AM34	D50	AR21	D30
AD32	V_{SS}	AK26	V_{SS}	AN5	A29	AR23	D36
AD34	V_{DD}	AK28	V_{DD}	AN7	D1	AR25	V_{SS}
AE1	A15	AK32	V_{SS}	AN9	V_{SS}	AR27	D40
AE3	A26	AK34	V_{DD}	AN11	D9	AR29	V_{DD}
AE5	V_{SS}	AL1	A23	AN13	D12	AR31	D43
AE31	V_{SS}	AL3	A25	AN15	V_{DD}		
AE33	D51	AL5	V_{SS}	AN17	D18		
AE35	D58	AL7	D3	AN19	D22		
AF2	A17	AL9	D4	AN21	V_{DD}		
AF4	A28	AL11	D10	AN23	D28		
AF32	D46	AL13	V_{SS}	AN25	D37		
AF34	D49	AL15	D16	AN27	V_{SS}		

GF Terminal Assignments – Alphabetical Listing

TERMINAL		TERMINAL		TERMINAL		TERMINAL	
NAME	NO.	NAME	NO.	NAME	NO.	NAME	NO.
A0	L1	CAS/DQM0	D20	D22	AN19	D61	U33
A1	G5	CAS/DQM1	B20	D23	AM20	D62	T32
A2	G3	CAS/DQM2	C19	D24	AL21	D63	W35
A3	H4	CAS/DQM3	B16	D25	AM22	DBEN	C25
A4	P2	CAS/DQM4	E17	D26	AP20	DDIN	B26
A5	T2	CAS/DQM5	A15	D27	AK22	DSF	A23
A6	K4	CAS/DQM6	B14	D28	AN23	EINT1	A31
A7	L3	CAS/DQM7	A13	D29	AL25	EINT2	G31
A8	N3	CBLNK0/VBLNK0	C31	D30	AR21	EINT3	E29
A9	P4	CBLNK1/VBLNK1	G33	D31	AM26	EMU0	T34
A10	U3	CLKIN	C17	D32	AL27	EMU1	J35
A11	W1	CLKOUT	E25	D33	AM28	FAULT	E3
A12	Y2	CSYNC0/HBLNK0	B32	D34	AP22	FCLK0	B28
A13	T4	CSYNC1/HBLNK1	H32	D35	AN29	FCLK1	D22
A14	AB2	CT0	C11	D36	AR23	FF1	U31
A15	AE1	CT1	A5	D37	AN25	FF2	AK18
A16	AG1	CT2	F18	D38	AL29	HACK	A9
A17	AF2	D0	AR5	D39	AP26	HREQ	E15
A18	W3	D1	AN7	D40	AR27	HSYNC0	E33
A19	Y4	D2	AM8	D41	AP28	HSYNC1	K34
A20	AH2	D3	AL7	D42	AJ33	LINT4	E27
A21	AB4	D4	AL9	D43	AR31	PS0	F14
A22	AC3	D5	AP8	D44	AH32	PS1	B8
A23	AL1	D6	AM10	D45	AN31	PS2	C13
A24	AM2	D7	AR9	D46	AF32	PS3	U5
A25	AL3	D8	AP10	D47	AP32	RAS	A21
A26	AE3	D9	AN11	D48	AL33	READY	E9
A27	AP4	D10	AL11	D49	AF34	REQ0	D16
A28	AF4	D11	AR13	D50	AM34	REQ1	B10
A29	AN5	D12	AN13	D51	AE33	RESET	D14
A30	AH4	D13	AP14	D52	AC33	RETRY	D2
A31	AJ3	D14	AM14	D53	AL35	RL	E19
AS0	D8	D15	AR15	D54	AH34	SCLK0	D28
AS1	E1	D16	AL15	D55	AB32	SCLK1	A27
AS2	C7	D17	AP16	D56	Y34	STATUS0	H2
BS0	E11	D18	AN17	D57	AG35	STATUS1	J1
BS1	B4	D19	AM16	D58	AE35	STATUS2	E7
CAREA0	C29	D20	AL17	D59	W33	STATUS3	C5
CAREA1	D26	D21	AL19	D60	AB34	STATUS4	K2

GF Terminal Assignments – Alphabetical Listing (Continued)

TERMINAL		TERMINAL		TERMINAL		TERMINAL	
NAME	NO.	NAME	NO.	NAME	NO.	NAME	NO.
STATUS5	E21	V_{DD}	R31	V_{DD}	AR29	V_{SS}	AA35
TCK	E35	V_{DD}	R33	V_{SS}	A11	V_{SS}	AC5
TDI	H34	V_{DD}	U1	V_{SS}	A19	V_{SS}	AC31
TDO	P32	V_{DD}	U35	V_{SS}	A25	V_{SS}	AD4
TMS	N33	V_{DD}	V2	V_{SS}	C3	V_{SS}	AD32
$\overline{TRG}/\overline{CAS}$	B22	V_{DD}	V34	V_{SS}	C9	V_{SS}	AE5
$\overline{TRST}$	L33	V_{DD}	AA3	V_{SS}	C27	V_{SS}	AE31
UTIME	D10	V_{DD}	AA5	V_{SS}	D6	V_{SS}	AG3
V_{DD}	A7	V_{DD}	AA31	V_{SS}	D12	V_{SS}	AG33
V_{DD}	A17	V_{DD}	AA33	V_{SS}	D18	V_{SS}	AJ5
V_{DD}	A29	V_{DD}	AC1	V_{SS}	D24	V_{SS}	AJ31
V_{DD}	B6	V_{DD}	AC35	V_{SS}	D30	V_{SS}	AK4
V_{DD}	B12	V_{DD}	AD2	V_{SS}	E5	V_{SS}	AK10
V_{DD}	B18	V_{DD}	AD34	V_{SS}	E13	V_{SS}	AK14
V_{DD}	B24	V_{DD}	AG5	V_{SS}	E23	V_{SS}	AK20
V_{DD}	B30	V_{DD}	AG31	V_{SS}	E31	V_{SS}	AK26
V_{DD}	C15	V_{DD}	AJ1	V_{SS}	F4	V_{SS}	AK32
V_{DD}	C21	V_{DD}	AJ35	V_{SS}	F10	V_{SS}	AL5
V_{DD}	D4	V_{DD}	AK2	V_{SS}	F16	V_{SS}	AL13
V_{DD}	D32	V_{DD}	AK8	V_{SS}	F22	V_{SS}	AL23
V_{DD}	F2	V_{DD}	AK12	V_{SS}	F26	V_{SS}	AL31
V_{DD}	F8	V_{DD}	AK16	V_{SS}	F32	V_{SS}	AM6
V_{DD}	F12	V_{DD}	AK24	V_{SS}	J3	V_{SS}	AM12
V_{DD}	F20	V_{DD}	AK28	V_{SS}	J33	V_{SS}	AM18
V_{DD}	F24	V_{DD}	AK34	V_{SS}	L5	V_{SS}	AM24
V_{DD}	F28	V_{DD}	AM4	V_{SS}	L31	V_{SS}	AM30
V_{DD}	F34	V_{DD}	AM32	V_{SS}	M4	V_{SS}	AN9
V_{DD}	G1	V_{DD}	AN15	V_{SS}	M32	V_{SS}	AN27
V_{DD}	G35	V_{DD}	AN21	V_{SS}	N5	V_{SS}	AR11
V_{DD}	J5	V_{DD}	AN33	V_{SS}	N31	V_{SS}	AR17
V_{DD}	J31	V_{DD}	AP6	V_{SS}	R1	V_{SS}	AR25
V_{DD}	M2	V_{DD}	AP12	V_{SS}	R35	$\overline{VSYNC0}$	D34
V_{DD}	M34	V_{DD}	AP18	V_{SS}	V4	$\overline{VSYNC1}$	K32
V_{DD}	N1	V_{DD}	AP24	V_{SS}	V32	$\overline{W}$	C23
V_{DD}	N35	V_{DD}	AP30	V_{SS}	W5	$\overline{XPT0}$	P34
V_{DD}	R3	V_{DD}	AR7	V_{SS}	W31	$\overline{XPT1}$	L35
V_{DD}	R5	V_{DD}	AR19	V_{SS}	AA1	$\overline{XPT2}$	Y32

GGP Terminal Assignments – Numerical Listing

TERMINAL		TERMINAL		TERMINAL		TERMINAL	
NO.	**NAME**	**NO.**	**NAME**	**NO.**	**NAME**	**NO.**	**NAME**
A1	No Connect	B1	No Connect	C1	No Connect	D1	A0
A2	No Connect	B2	No Connect	C2	No Connect	D2	V_{DD}
A3	No Connect	B3	No Connect	C3	No Connect	D3	STATUS4
A4	STATUS1	B4	STATUS2	C4	V_{SS}	D4	STATUS3
A5	AS1	B5	AS2	C5	STATUS0	D5	V_{DD}
A6	$\overline{RETRY}$	B6	READY	C6	$\overline{FAULT}$	D6	AS0
A7	CT1	B7	BS0	C7	BS1	D7	$\overline{UTIME}$
A8	PS0	B8	PS1	C8	PS2	D8	CT0
A9	V_{DD}	B9	$\overline{HACK}$	C9	$\overline{HREQ}$	D9	$\overline{RESET}$
A10	V_{SS}	B10	V_{SS}	C10	V_{SS}	D10	REQ1
A11	V_{SS}	B11	V_{DD}	C11	V_{DD}	D11	REQ0
A12	V_{DD}	B12	$\overline{CAS}$/DQM7	C12	CLKIN	D12	V_{DD}
A13	No Connect	B13	No Connect	C13	V_{SS}	D13	$\overline{CAS}$/DQM6
A14	No Connect	B14	V_{SS}	C14	$\overline{CAS}$/DQM5	D14	V_{DD}
A15	V_{DD}	B15	$\overline{CAS}$/DQM4	C15	$\overline{CAS}$/DQM3	D15	CT2
A16	$\overline{CAS}$/DQM2	B16	V_{SS}	C16	$\overline{CAS}$/DQM1	D16	V_{DD}
A17	$\overline{CAS}$/DQM0	B17	$\overline{RL}$	C17	V_{SS}	D17	V_{SS}
A18	V_{SS}	B18	$\overline{RAS}$	C18	V_{DD}	D18	$\overline{TRG}$/$\overline{CAS}$
A19	FCLK1	B19	V_{DD}	C19	$\overline{W}$	D19	STATUS5
A20	V_{DD}	B20	DSF	C20	V_{SS}	D20	$\overline{DBEN}$
A21	V_{DD}	B21	$\overline{DDIN}$	C21	CLKOUT	D21	CAREA1
A22	V_{SS}	B22	SCLK1	C22	V_{DD}	D22	FCLK0
A23	V_{SS}	B23	SCLK0	C23	V_{DD}	D23	CAREA0
A24	No Connect	B24	No Connect	C24	No Connect	D24	$\overline{LINT4}$
A25	No Connect	B25	No Connect	C25	No Connect	D25	$\overline{EINT3}$
A26	No Connect	B26	No Connect	C26	No Connect	D26	$\overline{EINT2}$

GGP Terminal Assignments – Numerical Listing (Continued)

TERMINAL		TERMINAL		TERMINAL		TERMINAL	
NO.	**NAME**	**NO.**	**NAME**	**NO.**	**NAME**	**NO.**	**NAME**
E1	A3	J23	$\overline{\text{HSYNC0}}$	P1	No Connect	V23	D52
E2	A2	J24	$\overline{\text{TRST}}$	P2	No Connect	V24	V_{SS}
E3	V_{SS}	J25	TCK	P3	V_{SS}	V25	V_{SS}
E4	A1	J26	TMS	P4	V_{SS}	V26	D53
E23	$\overline{\text{EINT1}}$	K1	A12	P23	V_{SS}	W1	A24
E24	$\overline{\text{CBLNK1}}$/$\overline{\text{VBLNK1}}$	K2	V_{DD}	P24	D59	W2	V_{SS}
E25	$\overline{\text{CBLNK0}}$/$\overline{\text{VBLNK0}}$	K3	A11	P25	V_{DD}	W3	V_{SS}
E26	V_{SS}	K4	V_{SS}	P26	No Connect	W4	V_{SS}
F1	A4	K23	TDI	R1	V_{SS}	W23	D49
F2	V_{DD}	K24	TDO	R2	A17	W24	D50
F3	V_{DD}	K25	EMU1	R3	A18	W25	D51
F4	V_{DD}	K26	$\overline{\text{XPT0}}$	R4	V_{SS}	W26	V_{DD}
F23	V_{SS}	L1	V_{DD}	R23	$\overline{\text{XPT2}}$	Y1	A25
F24	$\overline{\text{CSYNC1}}$/$\overline{\text{HBLNK1}}$	L2	A13	R24	D57	Y2	A26
F25	V_{DD}	L3	V_{SS}	R25	V_{DD}	Y3	A27
F26	$\overline{\text{CSYNC0}}$/$\overline{\text{HBLNK0}}$	L4	V_{SS}	R26	D58	Y4	V_{DD}
G1	V_{SS}	L23	$\overline{\text{XPT1}}$	T1	A19	Y23	V_{DD}
G2	V_{SS}	L24	V_{SS}	T2	V_{DD}	Y24	D48
G3	A5	L25	V_{SS}	T3	V_{DD}	Y25	V_{SS}
G4	V_{SS}	L26	EMU0	T4	A20	Y26	V_{SS}
G23	$\overline{\text{VSYNC1}}$	M1	V_{DD}	T23	V_{DD}	AA1	V_{DD}
G24	$\overline{\text{VSYNC0}}$	M2	A15	T24	V_{DD}	AA2	V_{DD}
G25	V_{SS}	M3	PS3	T25	D56	AA3	A28
G26	V_{SS}	M4	A14	T26	V_{SS}	AA4	V_{SS}
H1	A8	M23	V_{DD}	U1	V_{SS}	AA23	D45
H2	V_{DD}	M24	D63	U2	V_{SS}	AA24	D46
H3	A7	M25	D62	U3	A21	AA25	D47
H4	A6	M26	D61	U4	V_{DD}	AA26	V_{DD}
H23	$\overline{\text{HSYNC1}}$	N1	No Connect	U23	V_{DD}	AB1	V_{SS}
H24	V_{DD}	N2	A16	U24	D54	AB2	A29
H25	V_{DD}	N3	V_{DD}	U25	V_{SS}	AB3	A30
H26	V_{DD}	N4	V_{DD}	U26	D55	AB4	V_{SS}
J1	A10	N23	V_{SS}	V1	A22	AB23	V_{DD}
J2	V_{DD}	N24	D60	V2	A23	AB24	D44
J3	A9	N25	No Connect	V3	V_{DD}	AB25	V_{SS}
J4	V_{SS}	N26	No Connect	V4	V_{DD}	AB26	V_{SS}

GGP Terminal Assignments – Numerical Listing (Continued)

TERMINAL		TERMINAL		TERMINAL		TERMINAL	
NO.	NAME	NO.	NAME	NO.	NAME	NO.	NAME
AC1	A31	AD1	No Connect	AE1	No Connect	AF1	No Connect
AC2	V_{DD}	AD2	No Connect	AE2	No Connect	AF2	No Connect
AC3	V_{DD}	AD3	No Connect	AE3	No Connect	AF3	No Connect
AC4	D0	AD4	V_{SS}	AE4	D1	AF4	D2
AC5	D3	AD5	V_{DD}	AE5	D4	AF5	V_{SS}
AC6	V_{SS}	AD6	D5	AE6	D6	AF6	D7
AC7	V_{DD}	AD7	V_{DD}	AE7	D8	AF7	V_{SS}
AC8	D9	AD8	D10	AE8	D11	AF8	V_{DD}
AC9	D12	AD9	V_{SS}	AE9	D13	AF9	D14
AC10	V_{DD}	AD10	V_{DD}	AE10	D15	AF10	V_{SS}
AC11	D16	AD11	V_{SS}	AE11	V_{SS}	AF11	D17
AC12	D18	AD12	D19	AE12	V_{DD}	AF12	V_{DD}
AC13	D20	AD13	V_{SS}	AE13	V_{SS}	AF13	No Connect
AC14	V_{DD}	AD14	D21	AE14	No Connect	AF14	No Connect
AC15	D24	AD15	V_{DD}	AE15	D23	AF15	D22
AC16	D27	AD16	D26	AE16	D25	AF16	V_{SS}
AC17	V_{SS}	AD17	V_{SS}	AE17	D28	AF17	V_{DD}
AC18	D31	AD18	D30	AE18	V_{DD}	AF18	D29
AC19	V_{DD}	AD19	V_{DD}	AE19	D32	AF19	V_{SS}
AC20	D35	AD20	D34	AE20	D33	AF20	V_{SS}
AC21	D37	AD21	V_{SS}	AE21	D36	AF21	V_{DD}
AC22	D40	AD22	V_{DD}	AE22	D39	AF22	D38
AC23	D42	AD23	D41	AE23	V_{SS}	AF23	V_{SS}
AC24	D43	AD24	No Connect	AE24	No Connect	AF24	No Connect
AC25	V_{DD}	AD25	No Connect	AE25	No Connect	AF25	No Connect
AC26	V_{DD}	AD26	No Connect	AE26	No Connect	AF26	No Connect

GGP Terminal Assignments – Alphabetical Listing

TERMINAL		TERMINAL		TERMINAL		TERMINAL	
NAME	NO.	NAME	NO.	NAME	NO.	NAME	NO.
A0	D1	$\overline{\mathrm{CAS}}$/DQM0	A17	D22	AF15	D61	M26
A1	E4	$\overline{\mathrm{CAS}}$/DQM1	C16	D23	AE15	D62	M25
A2	E2	$\overline{\mathrm{CAS}}$/DQM2	A16	D24	AC15	D63	M24
A3	E1	$\overline{\mathrm{CAS}}$/DQM3	C15	D25	AE16	$\overline{\mathrm{DBEN}}$	D20
A4	F1	$\overline{\mathrm{CAS}}$/DQM4	B15	D26	AD16	$\overline{\mathrm{DDIN}}$	B21
A5	G3	$\overline{\mathrm{CAS}}$/DQM5	C14	D27	AC16	DSF	B20
A6	H4	$\overline{\mathrm{CAS}}$/DQM6	D13	D28	AE17	$\overline{\mathrm{EINT1}}$	E23
A7	H3	$\overline{\mathrm{CAS}}$/DQM7	B12	D29	AF18	$\overline{\mathrm{EINT2}}$	D26
A8	H1	$\overline{\mathrm{CBLNK0}}$/$\overline{\mathrm{VBLNK0}}$	E25	D30	AD18	$\overline{\mathrm{EINT3}}$	D25
A9	J3	$\overline{\mathrm{CBLNK1}}$/$\overline{\mathrm{VBLNK1}}$	E24	D31	AC18	EMU0	L26
A10	J1	CLKIN	C12	D32	AE19	EMU1	K25
A11	K3	CLKOUT	C21	D33	AE20	$\overline{\mathrm{FAULT}}$	C6
A12	K1	$\overline{\mathrm{CSYNC0/HBLNK0}}$	F26	D34	AD20	FCLK0	D22
A13	L2	$\overline{\mathrm{CSYNC1/HBLNK1}}$	F24	D35	AC20	FCLK1	A19
A14	M4	CT0	D8	D36	AE21	$\overline{\mathrm{HACK}}$	B9
A15	M2	CT1	A7	D37	AC21	$\overline{\mathrm{HREQ}}$	C9
A16	N2	CT2	D15	D38	AF22	$\overline{\mathrm{HSYNC0}}$	J23
A17	R2	D0	AC4	D39	AE22	$\overline{\mathrm{HSYNC1}}$	H23
A18	R3	D1	AE4	D40	AC22	$\overline{\mathrm{LINT4}}$	D24
A19	T1	D2	AF4	D41	AD23	PS0	A8
A20	T4	D3	AC5	D42	AC23	PS1	B8
A21	U3	D4	AE5	D43	AC24	PS2	C8
A22	V1	D5	AD6	D44	AB24	PS3	M3
A23	V2	D6	AE6	D45	AA23	$\overline{\mathrm{RAS}}$	B18
A24	W1	D7	AF6	D46	AA24	READY	B6
A25	Y1	D8	AE7	D47	AA25	REQ0	D11
A26	Y2	D9	AC8	D48	Y24	REQ1	D10
A27	Y3	D10	AD8	D49	W23	$\overline{\mathrm{RESET}}$	D9
A28	AA3	D11	AE8	D50	W24	$\overline{\mathrm{RETRY}}$	A6
A29	AB2	D12	AC9	D51	W25	$\overline{\mathrm{RL}}$	B17
A30	AB3	D13	AE9	D52	V23	SCLK0	B23
A31	AC1	D14	AF9	D53	V26	SCLK1	B22
AS0	D6	D15	AE10	D54	U24	STATUS0	C5
AS1	A5	D16	AC11	D55	U26	STATUS1	A4
AS2	B5	D17	AF11	D56	T25	STATUS2	B4
BS0	B7	D18	AC12	D57	R24	STATUS3	D4
BS1	C7	D19	AD12	D58	R26	STATUS4	D3
CAREA0	D23	D20	AC13	D59	P24	STATUS5	D19
CAREA1	D21	D21	AD14	D60	N24		

GGP Terminal Assignments – Alphabetical Listing (Continued)

TERMINAL		TERMINAL		TERMINAL		TERMINAL	
NAME	NO.	NAME	NO.	NAME	NO.	NAME	NO.
TCK	J25	V_{DD}	P25	V_{SS}	A10	V_{SS}	U25
TDI	K23	V_{DD}	R25	V_{SS}	A11	V_{SS}	V24
TDO	K24	V_{DD}	T2	V_{SS}	A18	V_{SS}	V25
TMS	J26	V_{DD}	T3	V_{SS}	A22	V_{SS}	W2
$\overline{TRG}$/$\overline{CAS}$	D18	V_{DD}	T23	V_{SS}	A23	V_{SS}	W3
$\overline{TRST}$	J24	V_{DD}	T24	V_{SS}	B10	V_{SS}	W4
$\overline{UTIME}$	D7	V_{DD}	U4	V_{SS}	B14	V_{SS}	Y25
V_{DD}	A9	V_{DD}	U23	V_{SS}	B16	V_{SS}	Y26
V_{DD}	A12	V_{DD}	V3	V_{SS}	C4	V_{SS}	AA4
V_{DD}	A15	V_{DD}	V4	V_{SS}	C10	V_{SS}	AB1
V_{DD}	A20	V_{DD}	W26	V_{SS}	C13	V_{SS}	AB4
V_{DD}	A21	V_{DD}	Y4	V_{SS}	C17	V_{SS}	AB25
V_{DD}	B11	V_{DD}	Y23	V_{SS}	C20	V_{SS}	AB26
V_{DD}	B19	V_{DD}	AA1	V_{SS}	D17	V_{SS}	AC6
V_{DD}	C11	V_{DD}	AA2	V_{SS}	E3	V_{SS}	AC17
V_{DD}	C18	V_{DD}	AA26	V_{SS}	E26	V_{SS}	AD4
V_{DD}	C22	V_{DD}	AB23	V_{SS}	F23	V_{SS}	AD9
V_{DD}	C23	V_{DD}	AC2	V_{SS}	G1	V_{SS}	AD11
V_{DD}	D2	V_{DD}	AC3	V_{SS}	G2	V_{SS}	AD13
V_{DD}	D5	V_{DD}	AC7	V_{SS}	G4	V_{SS}	AD17
V_{DD}	D12	V_{DD}	AC10	V_{SS}	G25	V_{SS}	AD21
V_{DD}	D14	V_{DD}	AC14	V_{SS}	G26	V_{SS}	AE11
V_{DD}	D16	V_{DD}	AC19	V_{SS}	J4	V_{SS}	AE13
V_{DD}	F2	V_{DD}	AC25	V_{SS}	K4	V_{SS}	AE23
V_{DD}	F3	V_{DD}	AC26	V_{SS}	L3	V_{SS}	AF5
V_{DD}	F4	V_{DD}	AD5	V_{SS}	L4	V_{SS}	AF7
V_{DD}	F25	V_{DD}	AD7	V_{SS}	L24	V_{SS}	AF10
V_{DD}	H2	V_{DD}	AD10	V_{SS}	L25	V_{SS}	AF16
V_{DD}	H24	V_{DD}	AD15	V_{SS}	N23	V_{SS}	AF19
V_{DD}	H25	V_{DD}	AD19	V_{SS}	P3	V_{SS}	AF20
V_{DD}	H26	V_{DD}	AD22	V_{SS}	P4	$\overline{V_{SS}}$	AF23
V_{DD}	J2	V_{DD}	AE12	V_{SS}	P23	$\overline{VSYNC0}$	G24
V_{DD}	K2	V_{DD}	AE18	V_{SS}	R1	$\overline{VSYNC1}$	G23
V_{DD}	L1	V_{DD}	AF8	V_{SS}	R4	$\overline{W}$	C19
V_{DD}	M1	V_{DD}	AF12	V_{SS}	T26	$\overline{XPT0}$	K26
V_{DD}	M23	V_{DD}	AF17	V_{SS}	U1	$\overline{XPT1}$	L23
V_{DD}	N3	V_{DD}	AF21	V_{SS}	U2	$\overline{XPT2}$	R23
V_{DD}	N4						

Terminal Functions

TERMINAL NAME	TYPE†	DESCRIPTION
LOCAL MEMORY INTERFACE		
A31–A0	O	Address bus. A31–A0 output the 32-bit byte address of the external memory cycle. The address can be multiplexed for DRAM accesses.
AS2–AS0	I	Address-shift selection. AS2–AS0 determine how the column address appears on the address bus. Eight shift values are supported, including zero.
BS1–BS0	I	Bus-size selection. BS1–BS0 indicate the bus size of the memory or other device being accessed, allowing dynamic bus sizing for data buses less than 64-bits wide.
CT2–CT0	I	Cycle-timing selection. CT2–CT0 signals determine the timing of the current memory access.
D63–D0	I/O	Data bus. D63–D0 transfer up to 64 bits of data per memory cycle into or out of the ’C80.
$\overline{\text{DBEN}}$	O	Data-buffer enable. $\overline{\text{DBEN}}$ drives the active-low output-enables of bi-directional transceivers that can be used to buffer input and output data on D63–D0.
$\overline{\text{DDIN}}$	O	Data-direction indicator. $\overline{\text{DDIN}}$ indicates the direction of the data that passes through the transceivers. When $\overline{\text{DDIN}}$ is low, the transfer is from external memory into the ’C80.
$\overline{\text{FAULT}}$	I	Fault. $\overline{\text{FAULT}}$ is driven low by external circuitry to inform the ’C80 that a fault has occurred on the current memory row-access.
PS3–PS0	I	Page-size indication. PS3–PS0 indicate the page size of the memory device(s) being accessed by the current cycle. The ’C80 uses this information to determine when to begin a new row-access.
READY	I	Ready. READY indicates that the external device is ready to complete the memory cycle. READY is driven low by external circuitry to insert wait states into a memory cycle.
$\overline{\text{RL}}$	O	Row latch. The high-to-low transition of $\overline{\text{RL}}$ can be used to latch the valid 32-bit byte address that is present on A31–A0.
$\overline{\text{RETRY}}$	I	Retry. $\overline{\text{RETRY}}$ is driven low by external circuitry to indicate that the addressed memory is busy. The ’C80 memory cycle is rescheduled.
STATUS5–STATUS0	O	Status code. At row time, STATUS5–STATUS0 indicate the type of cycle being performed. At column time, they identify the processor and type of request that initiated the cycle.
$\overline{\text{UTIME}}$	I	User-timing selection. $\overline{\text{UTIME}}$ causes the timing of $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$/DQM7–$\overline{\text{CAS}}$/DQM0 to be modified so that custom memory timings can be generated. During reset, $\overline{\text{UTIME}}$ selects the endian mode in which the ’C80 operates.
DRAM, VRAM, AND SDRAM CONTROL		
$\overline{\text{CAS}}$/DQM7– $\overline{\text{CAS}}$/DQM0	O	Column-address strobes. $\overline{\text{CAS}}$/DQM7–$\overline{\text{CAS}}$/DQM0 drive the $\overline{\text{CAS}}$ inputs of DRAMs and VRAMs, or the DQM input of SDRAMs. The eight strobes provide byte-write access to memory.
DSF	O	Special function. DSF selects special VRAM functions such as block-write, load color register, split-register transfer, and SGRAM block write.
$\overline{\text{RAS}}$	O	Row-address strobe. $\overline{\text{RAS}}$ drives the $\overline{\text{RAS}}$ inputs of DRAMs, VRAMs, and SDRAMs.
$\overline{\text{TRG}}$/$\overline{\text{CAS}}$	O	Transfer/output enable or column-address strobe. $\overline{\text{TRG}}$/$\overline{\text{CAS}}$ is used as an output-enable for DRAMs and VRAMs, and also as a transfer-enable for VRAMs. $\overline{\text{TRG}}$/$\overline{\text{CAS}}$ also drives the $\overline{\text{CAS}}$ inputs of SDRAMs.
$\overline{\text{W}}$	O	Write enable. $\overline{\text{W}}$ is driven low before $\overline{\text{CAS}}$ during write cycles. $\overline{\text{W}}$ controls the direction of the transfer during VRAM transfer cycles.

† I = input, O = output, Z = high impedance

Terminal Functions (Continued)

TERMINAL NAME	TYPE†	DESCRIPTION
HOST INTERFACE		
HACK	O	Host acknowledge. The ’C80 drives HACK output low following an active HREQ to indicate that it has driven the local-memory-bus signals to the high-impedance state and is relinquishing the bus. HACK is driven high asynchronously following HREQ being detected inactive, and then the ’C80 resumes driving the bus.
HREQ	I	Host request. An external device drives HREQ low to request ownership of the local-memory bus. When HREQ is high, the ’C80 owns and drives the bus. HREQ is synchronized internally to the ’C80’s internal clock. Also, HREQ is used at reset to determine the power-up state of the MP. If HREQ is low at the rising edge of RESET, the MP comes up running. If HREQ is high, the MP remains halted until the first interrupt occurrence on EINT3.
REQ1, REQ0	O	Internal cycle request. REQ1 and REQ0 provide a two-bit code indicating the highest-priority memory-cycle request that is being received by the TC. External logic can monitor REQ1 and REQ0 to determine if it is necessary to relinquish the local-memory bus to the ’C80.
SYSTEM CONTROL		
CLKIN	I	Input clock. CLKIN generates the internal ’C80 clocks to which all processor functions (except the frame timers) are synchronous.
CLKOUT	O	Local output clock. CLKOUT provides a way to synchronize external circuitry to internal timings. All ’C80 output signals (except the VC signals) are synchronous to this clock.
EINT1, EINT2, EINT3	I	Edge-triggered interrupts. EINT1, EINT2 and EINT3 allow external devices to interrupt the master processor (MP) on one of three interrupt levels (EINT1 is the highest priority). The interrupts are rising-edge triggered. EINT3 also serves as an unhalt signal. If the MP is powered-up halted, the first rising edge on EINT3 causes the MP to unhalt and fetch its reset vector (the EINT3 interrupt-pending bit is not set in this case).
LINT4	I	Level-triggered interrupt. LINT4 provides an active-low level-triggered interrupt to the MP. Its priority falls below that of the edge-triggered interrupts. Any interrupt request should remain low until it is recognized by the ’C80.
RESET	I	Reset. RESET is driven low to reset the ’C80 (all processors). During reset, all internal registers are set to their initial state and all outputs are driven to their inactive or high-impedance levels. During the rising edge of RESET, the MP reset mode and the ’C80’s operating endian mode are determined by the levels of HREQ and UTIME pins, respectively.
XPT2 – XPT0	I	External packet transfer. XPT2 – XPT0 are used by external devices to request a high-priority XPT by the TC.
EMULATION CONTROL		
EMU0, EMU1‡	I/O	Emulation pins. EMU0 and EMU1 are used to support emulation host interrupts, special functions targeted at a single processor, and multiprocessor halt-event communications.
TCK‡	I	Test clock. TCK provides the clock for the ’C80 IEEE-1149.1 logic, allowing it to be compatible with other IEEE-1149.1 devices, controllers, and test equipment designed for different clock rates.
TDI‡	I	Test data input. TDI provides input data for all IEEE-1149.1 instructions and data scans of the ’C80.
TDO	O	Test data output. TDO provides output data for all IEEE-1149.1 instructions and data scans of the ’C80.
TMS‡	I	Test-mode select. TMS controls the IEEE-1149.1 state machine.
TRST§	I	Test reset. TRST resets the ’C80 IEEE-1149.1 module. When low, all boundary-scan logic is disabled, allowing normal ’C80 operation.

† I = input, O = output, Z = high impedance
‡ This pin has an internal pullup and can be left unconnnected during normal operation.
§ This pin has an internal pulldown and can be left unconnnected during normal operation.

Terminal Functions (Continued)

TERMINAL NAME	TERMINAL TYPE†	DESCRIPTION
VIDEO INTERFACE		
CAREA0, CAREA1	O	Composite area. CAREA0 and CAREA1 define a special area such as an overscan boundary. This area represents the logical OR of the internal horizontal and vertical area signals.
CBLNK0 / VBLNK0, CBLNK1 / VBLNK1	O	Composite blanking / vertical blanking. Each of CBLNK0 / VBLNK0 and VBLNK1 provides one of two blanking functions, depending on the configuration of the CSYNC / HBLNK pin: Composite blanking disables pixel display / capture during both horizontal and vertical retrace periods and is enabled when CSYNC is selected for composite sync video systems. Vertical blanking disables pixel display / capture during vertical retrace periods and is enabled when HBLNK is selected for separate-sync video systems. Following reset, CBLNK0 / VBLNK0 and CBLNK1 / VBLNK1 are configured as CBLNK0 and CBLNK1, respectively.
CSYNC0 / HBLNK0, CSYNC1 / HBLNK1	I / O / Z	Composite sync / horizontal blanking. CSYNC0 / HBLNK0 and CSYNC1 / HBLNK1 can be programmed for one of two functions: Composite sync is for use on composite-sync video systems and can be programmed as an input, output, or high-impedance signal. As an input, the ’C80 extracts horizontal and vertical sync information from externally generated active-low sync pulses. As an output, the active-low composite sync pulses are generated from either external HSYNC and VSYNC signals or the ’C80’s internal video timers. In the high-impedance state, the pin is neither driven nor allowed to drive circuitry. Horizontal blank disables pixel display / capture during horizontal retrace periods in separate-sync video systems and can be used as an output only. Immediately following reset, CSYNC0 / HBLNK0 and CSYNC1 / HBLNK1 are configured as high-impedance CSYNC0 and CSYNC1, respectively.
FCLK0, FCLK1	I	Frame clock. FCLK0 and FCLK1 are derived from the external video system’s dotclock and are used to drive the ’C80 video logic for frame timer 0 and frame timer 1.
HSYNC0, HSYNC1	I / O / Z	Horizontal sync. HSYNC0 and HSYNC1 control the video system. They can be programmed as input, output, or high impedance signals. As an input, HSYNC synchronizes the video timer to externally generated horizontal sync pulses. As an output, HSYNC is an active-low horizontal sync pulse generated by the ’C80 on-chip frame timer. In the high-impedance state, the pin is not driven, and no internal synchronization is allowed to occur. Immediately following reset, HSYNC0 and HSYNC1 are in the high-impedance state.
SCLK0, SCLK1	I	Serial-data clock. SCLK0 and SCLK1 are used by the ’C80 SRT controller to track the VRAM tap point when using midline reload. SCLK0 and SCLK1 should be the same signals that clock the serial register on the VRAMs controlled by frame timer 0 and frame timer 1, respectively.
VSYNC0, VSYNC1	I / O / Z	Vertical sync. VSYNC0 and VSYNC1 control the video system. They can be programmed as inputs, outputs, or high-impedance signals. As inputs, VSYNCx synchronizes the frame timer to externally generated vertical-sync pulses. As outputs, VSYNCx are active-low vertical-sync pulses generated by the ’C80 on-chip frame timer. In the high-impedance state, the pin is not driven and no internal synchronization is allowed to occur. Immediately following reset, VSYNCx is in the high-impedance state.
POWER		
V_{SS}‡	I	Ground. Electrical ground inputs
V_{DD}‡	I	Power. Nominal 3.3-V power supply inputs
MISCELLANEOUS		
No Connect		No connect serves as an alignment key and must be left unconnected.
FF2 – FF1		FF2 – FF1 (GF package only) are reserved for factory use and should be left unconnected.

† I = input, O = output, Z = high-impedance
‡ For proper operation, all V_{DD} and V_{SS} pins must be connected externally.

architecture

Figure 1 shows the major components of the ’C80: the master processor (MP), the parallel digital signal processors (PPs), the transfer controller (TC), the video controller (VC), and the IEEE-1149.1 emulation interface. Shared access to on-chip RAM is achieved through the crossbar. Crossbar connections are represented by ○. Each PP can perform three accesses per cycle through its local (L), global (G), and instruction (I) ports. The MP can access two RAMs per cycle through its crossbar/data (C/D) and instruction (I) ports, and the TC can access one RAM through its crossbar interface. Up to 15 simultaneous accesses are supported in each cycle. Addresses can be changed every cycle, allowing the crossbar matrix to be changed on a cycle-by-cycle basis. Contention between processors for the same RAM in the same cycle is resolved by a round-robin priority scheme. In addition to the crossbar, a 32-bit datapath exists between the MP and the TC and VC. This allows the MP to access TC and VC on-chip registers that are memory mapped into the MP memory space.

The ’C80 has a 4G-byte address space as shown in Figure 2. The lower 32M bytes are used to address internal RAM and memory-mapped registers.

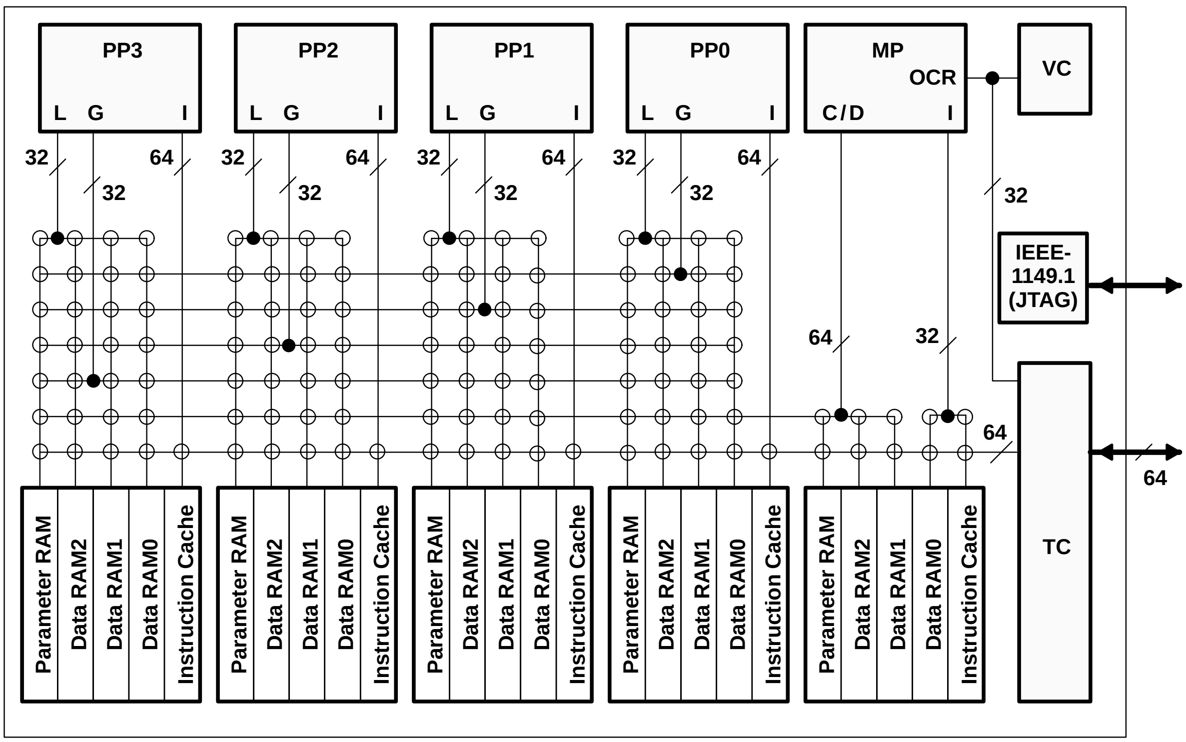


Figure 1. Block Diagram Showing Datapaths

architecture (continued)

Region	Start Address	End Address
PP0 Data RAM0 (2K Bytes)	0x00000000	0x000007FF
PP0 Data RAM1 (2K Bytes)	0x00000800	0x00000FFF
PP1 Data RAM0 (2K Bytes)	0x00001000	0x000017FF
PP1 Data RAM1 (2K Bytes)	0x00001800	0x00001FFF
PP2 Data RAM0 (2K Bytes)	0x00002000	0x000027FF
PP2 Data RAM1 (2K Bytes)	0x00002800	0x00002FFF
PP3 Data RAM0 (2K Bytes)	0x00003000	0x000037FF
PP3 Data RAM1 (2K Bytes)	0x00003800	0x00003FFF
Reserved (16K Bytes)	0x00004000	0x00007FFF
PP0 Data RAM2 (2K Bytes)	0x00008000	0x000087FF
Reserved (2K Bytes)	0x00008800	0x00008FFF
PP1 Data RAM2 (2K Bytes)	0x00009000	0x000097FF
Reserved (2K Bytes)	0x00009800	0x00009FFF
PP2 Data RAM2 (2K Bytes)	0x0000A000	0x0000A7FF
Reserved (2K Bytes)	0x0000A800	0x0000AFFF
PP3 Data RAM2 (2K Bytes)	0x0000B000	0x0000B7FF
Reserved (16 730 112 Bytes)	0x0000B800	0x00FFFFFF
PP0 Parameter RAM (2K Bytes)	0x01000000	0x010007FF
Reserved (2K Bytes)	0x01000800	0x01000FFF
PP1 Parameter RAM (2K Bytes)	0x01001000	0x010017FF
Reserved (2K Bytes)	0x01001800	0x01001FFF
PP2 Parameter RAM (2K Bytes)	0x01002000	0x010027FF
Reserved (2K Bytes)	0x01002800	0x01002FFF
PP3 Parameter RAM (2K Bytes)	0x01003000	0x010037FF

Region	Start Address	End Address
Reserved (51 200 Bytes)	0x01003800	0x0100FFFF
MP Parameter RAM (2K Bytes)	0x01010000	0x010107FF
Reserved (8 327 168 Bytes)	0x01010800	0x018017FF
PP0 Instruction Cache (2K Bytes)	0x01801800	0x01801FFF
Reserved (6K Bytes)	0x01802000	0x018037FF
PP1 Instruction Cache (2K Bytes)	0x01803800	0x01803FFF
Reserved (6K Bytes)	0x01804000	0x018057FF
PP2 Instruction Cache (2K Bytes)	0x01805800	0x01805FFF
Reserved (6K Bytes)	0x01806000	0x018077FF
PP3 Instruction Cache (2K Bytes)	0x01807800	0x01807FFF
Reserved (32K Bytes)	0x01808000	0x0180FFFF
MP Data Cache (4K Bytes)	0x01810000	0x01810FFF
Reserved (28K Bytes)	0x01811000	0x01817FFF
MP Instruction Cache (4K Bytes)	0x01818000	0x01818FFF
Reserved (28K Bytes)	0x01819000	0x0181FFFF
Memory-Mapped TC Registers (512 Bytes)	0x01820000	0x018201FF
Memory-Mapped VC Registers (512 Bytes)	0x01820200	0x018203FF
Reserved (8 327 168 Bytes)	0x01820400	0x01FFFFFF
External Memory (4 064M Bytes)	0x02000000	0xFFFFFFFF

Figure 2. Memory Map

master processor (MP) architecture

The master processor (MP) is a 32-bit RISC processor with an integral IEEE-754 floating-point unit. The MP is designed for effective execution of C code and is capable of performing at well over 130K dhrystones/s. Major tasks which the MP typically performs are:

- Task control and user interface
- Information processing and analysis
- IEEE-754 floating point (including graphics transforms)

MP functional block diagram

Figure 3 shows a block diagram of the master processor. Key features of the MP include:

- 32-bit RISC processor
  - Load/store architecture
  - Three operand arithmetic and logical instructions
- 4K-byte instruction cache and 4K-byte data cache
  - Four-way set associative
  - LRU replacement
  - Data writeback
- 2K-byte non-cached parameter RAM
- Thirty-one 32-bit general-purpose registers
- Register and accumulator scoreboard
- 15-bit or 32-bit immediate constants
- 32-bit byte addressing
- Scalable timer
- Leftmost-one and rightmost-one logic
- IEEE-754 floating-point hardware
  - Four double-precision floating-point vector accumulators
  - Vector floating-point instructions
    Floating-point operation and parallel load or store
    Multiply and accumulate
- High performance
  - 60 million instructions per second (MIPS)
  - 120 million floating-point operations per second (MFLOPS)
  - Over 130K dhrystones/s

MP functional block diagram (continued)

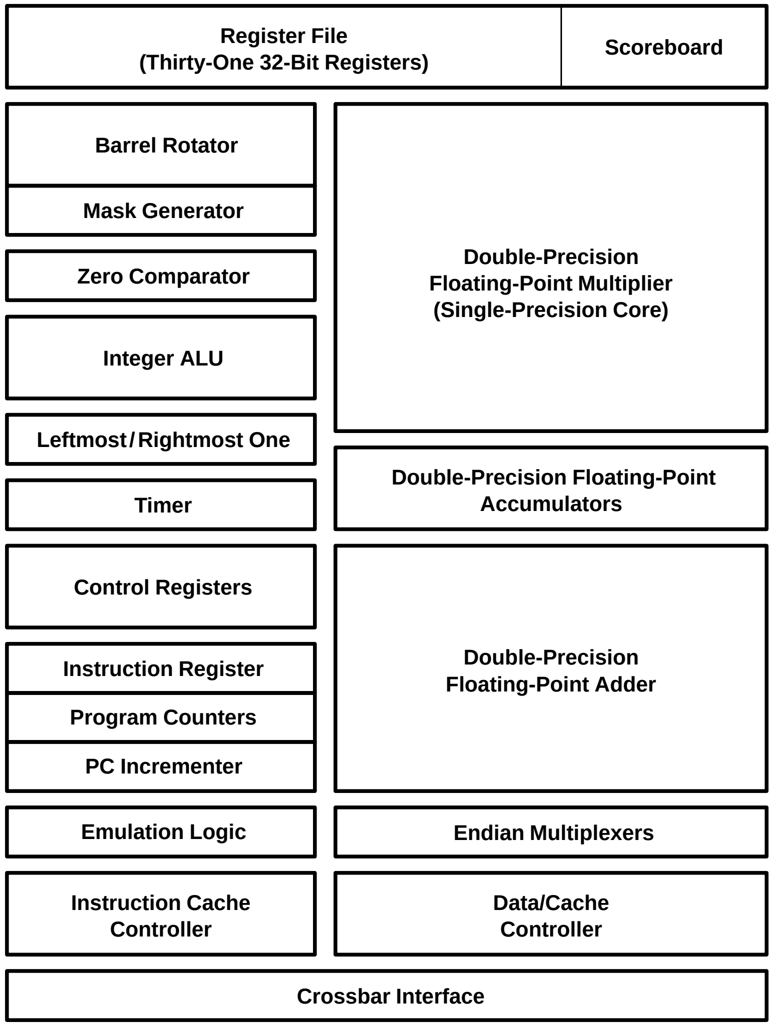


Figure 3. MP Block Diagram

MP general-purpose registers

The MP contains 31 32-bit general-purpose registers, R1–R31. Register R0 always reads as zero and writes to it are discarded. Double precision values are always stored in an even-odd register pair with the higher numbered register always holding the sign bit and exponent. The R0/R1 pair is not available for this use. A scoreboard keeps track of which registers are awaiting loads or the result of a previous instruction and stalls the instruction pipeline until the register contains valid data. As a recommended software convention, typically R1 is used as a stack pointer and R31 as a return-address link register.

Figure 4 shows the MP general-purpose registers.

MP general-purpose registers (continued)

32-Bit Registers	64-Bit Register Pairs
Zero/Discard	Not Available
R1	
R2	R2, R3
R3	
R4	R4, R5
R5	
• • •	• • •
R30	R30, R31
R31	

Figure 4. MP General-Purpose Registers

The 32-bit registers can contain signed-integer, unsigned-integer, or single precision floating-point values. Signed and unsigned bytes and halfwords are sign extended or zero-filled. Doublewords may be stored in a 64-bit even/odd register pair. Double-precision floating-point values are referenced using the even register number or the register pair. Figure 5 through Figure 7 show the register data formats.

Format	31	30–23	22 … 0
Single Precision Floating Point	S	E E E E E E E E	M M M M M M M M M M M M M M M M M M M M M M M (MS … LS)
Signed 32-bit Integer	S (MS)	I I I I I I I I	I I I I I I I I I I I I I I I I I I I I I I I (LS)
Unsigned 32-Bit Integer	U (MS)	U U U U U U U U	U U U U U U U U U U U U U U U U U U U U U U U (LS)

Figure 5. MP Register 32-Bit Data Formats

MP general-purpose registers (continued)

Register	Bits 31 … 8	Bits 7 (MS) … 0 (LS)
Signed Byte	S S S S S S S S S S S S S S S S S S S S S S S S	S I I I I I I I
Unsigned Byte	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	U U U U U U U U

Register	Bits 31 … 16	Bits 15 (MS) … 0 (LS)
Signed Halfword	S S S S S S S S S S S S S S S S	S I I I I I I I I I I I I I I I
Unsigned Halfword	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	U U U U U U U U U U U U U U U U

Figure 6. MP Register 8-Bit and 16-Bit Data

Register	Bits 31 (MS) … 0
Odd Register	Most Significant 32-Bit Word

Register	Bits 31 … 0 (LS)
Even Register	Least Significant 32-Bit Word

Register	Bit 31	Bits 30 … 20	Bits 19 (MS) … 0
Double-Precision Floating-Point Odd Register	S	E E E E E E E E E E E	M M M M M M M M M M M M M M M M M M M M

Register	Bits 31 … 0 (LS)
Double-Precision Floating-Point Even Register	M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M

Figure 7. MP Register 64-Bit Data

MP double-precision floating-point accumulators

There are four double-precision floating-point registers (see Figure 8) to accumulate intermediate floating-point results.

	64 (MSB) … 0 (LSB)
a0	Accumulator 0
a1	Accumulator 1
a2	Accumulator 2
a3	Accumulator 3

Figure 8. Double-Precision Floating-Point Accumulators

MP control registers

In addition to the general-purpose registers, there are a number of control registers that are used to represent the state of the processor. Table 1 shows the control register numbers of the accessible registers.

Table 1. Control Register Numbers

NO.	NAME	DESCRIPTION	NO.	NAME	DESCRIPTION
0x0000	EPC	Exception Program Counter	0x0015–0x001F	—	Reserved
0x0001	EIP	Exception Instruction Pointer	0x0020	SYSSTK	System Stack Pointer
0x0002	CONFIG	Configuration	0x0021	SYSTMP	System Temporary Register
0x0003	—	Reserved	0x0022–0x002F	—	Reserved
0x0004	INTPEN	Interrupt Pending	0x0030	MPC	Emulator Exception Program Cntr
0x0005	—	Reserved	0x0031	MIP	Emulator Exception Instruction Ptr
0x0006	IE	Interrupt Enable	0x0032	—	Reserved
0x0007	—	Reserved	0x0033	ECOMCNTL	Emulator Communication Control
0x0008	FPST	Floating-Point Status	0x0034	ANASTAT	Emulation Analysis Status Reg
0x0009	—	Reserved	0x0035–0x0038	—	Reserved
0x000A	PPERROR	PP Error Indicators	0x0039	BRK1	Emulation Breakpoint 1 Reg.
0x000B	—	Reserved	0x003A	BRK2	Emulation Breakpoint 2 Reg.
0x000C	—	Reserved	0x003B–0x01FF	—	Reserved
0x000D	PKTREQ	Packet Request Register	0x0200 – 0x020F	ITAG0 – 15	Instruction Cache Tags 0 to 15
0x000E	TCOUNT	Current Counter Value	0x0300	ILRU	Instruction Cache LRU Register
0x000F	TSCALE	Counter Reload Value	0x0400–0x040F	DTAG0 – 15	Data Cache Tags 0 to 15
0x0010	FLTOP	Faulting Operation	0x0500	DLRU	Data Cache LRU Register
0x0011	FLTADR	Faulting Address	0x4000	IN0P	Vector Load Pointer 0
0x0012	FLTTAG	Faulting Tag	0x4001	IN1P	Vector Load Pointer 1
0x0013	FLTDTL	Faulting Data (low)	0x4002	OUTP	Vector Store Pointer
0x0014	FLTDTH	Faulting Date (high)			

MP pipeline registers

The MP uses a three-stage fetch, execute, access (FEA) pipeline. The primary pipeline registers are manipulated implicitly by branch and trap instructions and are not accessible by the user. The exception and emulation pipeline registers are user accessible as control registers. All pipeline registers are 32 bits.

	Program Execution Mode		
	Normal	Exception	Emulation
Program Counter	PC	EPC	MPC
Instruction Pointer	IP	EIP	MIP
Instruction Register	IR		

- Instruction register (IR) contains the instruction being executed.
- Instruction pointer (IP) points to the instruction being executed.
- Program counter (PC) points to the instruction being fetched.
- Exception/emulator instruction pointer (EIP/MIP) points to the instruction that would have been executed had the exception / emulation trap not occurred.
- Exception/emulator program counter (EPC/MPC) points to the instruction to be fetched on returning from the exception/emulation trap.

Figure 9. MP FEA Pipeline Registers

configuration (CONFIG) register (0x0002)

The CONFIG register controls or reflects the state of certain options as shown in Figure 10.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E	R	T	H	X	Reserved											Type				Reserved				Release				Reserved			

E Endian mode; 0 = big-endian, 1 = little-endian, read only
R PPData RAM round robin; 0 = fixed, 1 = variable, read / write
T TC PT round robin; 0 = variable, 1 = fixed, read / write.
H High priority MP events; 0 = disabled, 1 = enabled, read / write
X Externally initiated packet transfers; 0 = disabled, 1 = enabled, read / write
Type Number of PPs in device, read only
Release TMS320C80 version number

Figure 10. CONFIG Register

interrupt-enable (IE) register (0x0006)

The IE register contains enable bits for each of the interrupts/traps as shown in Figure 11. The global-interrupt-enable (ie) bit and the appropriate individual interrupt-enable bit must be set in order for an interrupt to occur.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
pe	x4	x3	bp	pb	pc	mi						p3	p2	p1	p0	io	mf		x2	x1	ti	f1	f0	fx	fu	fo		fz	fi		ie

pe PP error
x4 External interrupt 4 ($\overline{\text{LINT4}}$)
x3 External interrupt 3 ($\overline{\text{EINT3}}$)
bp Bad packet transfer
pb Packet transfer busy
pc Packet transfer complete
mi Message (MP self) interrupt
p3 PP3 message interrupt
p2 PP2 message interrupt
p1 PP1message interrupt
p0 PP0 message interrupt
io Integer overflow
mf Memory fault
x2 External interrupt 2 ($\overline{\text{EINT2}}$)
x1 External interrupt 1 ($\overline{\text{EINT1}}$)
ti MP timer interrupt
f1 Frame-timer 1 interrupt
f0 Frame-timer 0 interrupt
fx Floating-point inexact
fu Floating-point underflow
fo Floating-point overflow
fz Floating-point divide-by-zero
fi Floating-point invalid
ie Global-interrupt enable

Figure 11. IE Register

interrupt-pending (INTPEN) register (0x0004)

The bits in INTPEN register show the current state of each interrupt/trap. Pending interrupts do not occur unless the ie bit and corresponding interrupt-enable bit are set. Software must write a 1 to the appropriate INTPEN bit to clear an interrupt. Figure 12 shows the INTPEN register locations.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
pe	x4	x3	bp	pb	pc	mi						p3	p2	p1	p0	io	mf		x2	x1	ti	f1	f0	fx	fu	fo		fz	fi		

Figure 12. INTPEN Register

floating-point status register (FPST) (0x0008)

FPST contains status and control information for the FPU as shown in Figure 13. Bits 17–21 are read/write floating-point unit (FPU) control bits. Bits 22–26 are read/write accumulated status bits. All other bits show the status of the last FPU instruction to complete and are read only.

31–27	26	25	24	23	22	21	20	19	18–17	16–13	12	11	10–9	8–7	6	5	4	3	2	1	0
destination	ai	az	ao	au	ax	sm	fs	vm	drm	opcode	e1	e0	pd	rm		mo	i	z	o	u	x

dest	**Destination register value**
ai	**Accumulated value invalid**
az	**Accumulated divide-by-zero**
ao	**Accumulated overflow**
au	**Accumulated underflow**
ax	**Accumulated inexact**
sm	**Sequential mode select**
fs	**Floating-point stall**
vm	**Vector fast mode**
drm	**Rounding mode** **00 – nearest** **10 – positive ∞** **01 – zero** **11 – negative ∞**
opcode	**Last opcode**
e1	**The tenth MSB of exponent**
e0	**The ninth MSB of exponent**
pd	**Destination precision** **00 – single float** **10 – signed int** **01 – double float** **11 – unsigned int**
rm	**Rounding mode** **00 – nearest** **10 – positive ∞** **01 – zero** **11 – negative ∞**
mo	**Int multiply overflow**
i	**Invalid**
z	**Divide-by-zero**
o	**Overflow**
u	**Underflow**
x	**Inexact**

Figure 13. FPSTS Register

PP error register (PPERROR) (0x000A)

The bits in the PPERROR register reflect parallel processor errors (see Figure 14). The MP can use these when a PP error interrupt occurs to determine the cause of the error.

31–20	19	18	17	16	15–12	11	10	9	8	7–4	3	2	1	0
Reserved	**h**	**h**	**h**	**h**	**Reserved**	**i**	**i**	**i**	**i**	**Reserved**	**f**	**f**	**f**	**f**
PP#	3	2	1	0	PP#	3	2	1	0	PP#	3	2	1	0

h – PP halted **i – PP illegal instruction** **f – PP fault type; 0 = icache, 1 = DEA**

Figure 14. PPERROR Register

packet-transfer request register (PKTREQ) (0x000D)

PKTREQ controls the submission and priority of packet-transfer requests as shown in Figure 15. It also indicates that a packet transfer is currently active.

31–5	4	3	2	1	0
Reserved	**I**	**F**	**S**	**Q**	**P**

I – Immediate (urgent) priority selected
F – High (foreground) priority selected
S – Suspend packet transfer
Q – Packet transfer queued; read only
P – Submit packet-transfer request

Figure 15. PKTREQ Register

memory-fault registers

The five read-only memory-fault registers contain information about memory address exceptions, as shown in Figure 16.

FLTOP (0x0010)

Bits	31–27	26–18	17–16	15	14–13	12	11	10	9	8–4	3–0
Field	Dest	Reserved	K		SZ	i	d	x	R	Reserved	Block

FLTTAG (0x0012)

Bits	31–10	9	8	7	6	5	4	3	2	1–0
Field	22-Bit Cache Tag Address	P	D	P	D	P	D	P	D	
Sub-Block		3		2		1		0		

Register	31 – 0
FLTADR (0x0011)	Faulting Address Accessed by the Instruction
FLTDTH (0x0013)	Faulting Write Most-Significant-Data Word
FLTDTL (0x0014)	Faulting Write Least-Significant-Data Word

Dest Destination Register Number
K Kind of Operation:
- 00 – load
- 01 – unsigned load
- 10 – store
- 11 – cache flush/clean

SZ Size of Data:
- 00 – 8-bit
- 01 – 16-bit
- 10 – 32-bit
- 11 – 64-bit

i MP icache fault
d MP dcache fault
x DEA Fault
R Modified return sequence
Block Faulting block number
P Sub-block is present.
D Dirty bit set

Figure 16. Memory-Fault Registers

MP cache registers

The ILRU and DLRU registers track least-recently-used (LRU) information for the sixteen instruction-cache and sixteen data-cache blocks. The ITAGxx registers contain block addresses and the present flags for each sub-block. DTAGxx registers are identical to ITAGxx registers but include dirty bits for each sub-block. Figure 17 shows the cache registers.

ILRU (0x0300)
DLRU (0x0500)

31–30	29–28	27–26	25–24	23–22	21–20	19–18	17–16	15–14	13–12	11–10	9–8	7–6	5–4	3–2	1–0
mru	nmru	nlru	lru	mru	nmru	nlru	lru	mru	nmru	nlru	lru	mru	nmru	nlru	lru
Set 3				Set 2				Set 1				Set 0			

ITAG0 – ITAG15 (0x0200 – 0x020F)

31–10	9	8	7	6	5	4	3	2	1–0
22-Bit Cache Tag Address	P		P		P		P		
	3		2		1		0		

Sub-Block

DTAG0 – DTAG15 (0x0400 – 0x040F)

31–10	9	8	7	6	5	4	3	2	1–0
22-Bit Cache Tag Address	P	D	P	D	P	D	P	D	
	3		2		1		0		

Sub-Block

mru	Most-recently-used block	lru	Least-recently-used block
nmru	Next most-recently-used block	P	Sub-block present
nlru	Next least-recently-used block	D	Sub-block dirty

mru, nmru, nlru, and lru have the value 0, 1, 2, or 3 representing the block number and are mutually exclusive for each set.

Figure 17. Cache Registers

MP cache architecture

The MP contains two four-way set-associative, 4K caches for instructions and data. Each cache is divided into four sets with four blocks in each set. Each block represents 256 bytes of contiguous instructions or data and is aligned to a 256-byte address boundary. Each block is partitioned into four sub-blocks that each contain sixteen 32-bit words and are aligned to 64-byte boundaries within the block. Cache misses cause one sub-block to be loaded into cache. Figure 18 shows the cache architecture for one of the four sets in each cache. Figure 19 shows how addresses map into the cache using the cache tags and address bits.

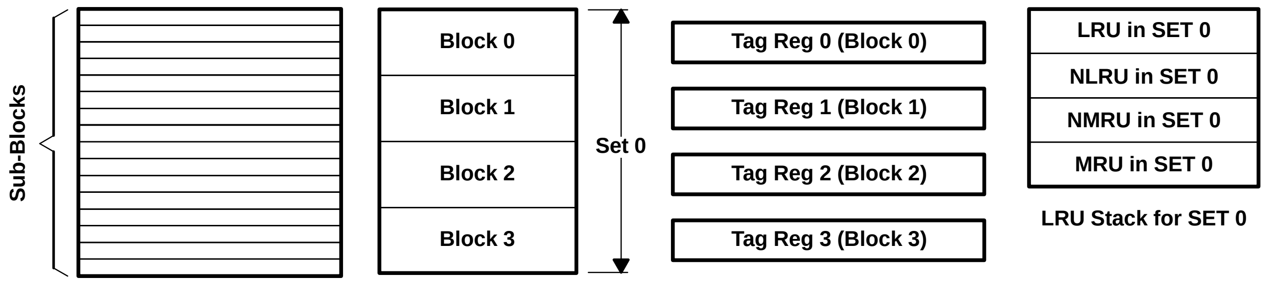
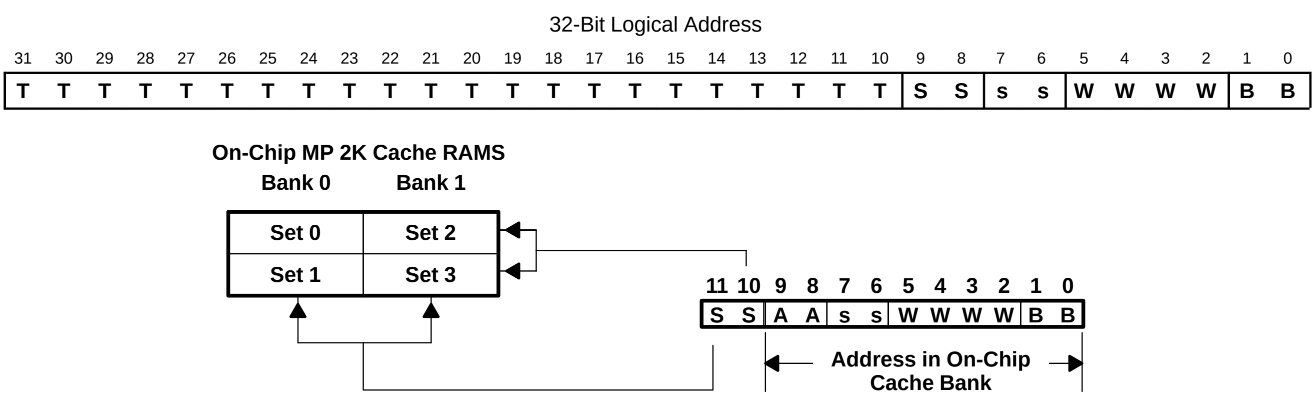


Figure 18. MP Cache Architecture (x4 Sets)



T – Tag Address Bits
S – Set Select Bits (0–3)
s – Sub-Block (within block) Select (0–3)
W – Word (within sub-block) Select (0–15)
B – Byte (within word) Select (0–3)
A – Block Select (which tag matched) (0–3)

Figure 19. MP Cache Addressing

MP parameter RAM

The parameter RAM is a noncachable, 2K-byte, on-chip RAM which contains MP-interrupt vectors, MP-requested TC task buffers, and a general-purpose area. Figure 20 shows the parameter RAM address map.

Address	Contents
0x01010000 – 0x0101007F	Suspended PT Parameters (128 Bytes)
0x01010080 – 0x010100DF	Reserved (96 Bytes)
0x010100E0 – 0x010100FB	XPT Linked List Start Addresses (28 Bytes)
0x010100FC – 0x010100FF	MP Linked List Start Address
0x01010100 – 0x0101017F	Off-Chip to Off-Chip PT Buffer (128 Bytes)
0x01010180 – 0x0101021F	Interrupt and Trap Vectors (160 Bytes)
0x01010220 – 0x0101029F	XPT Off-Chip to Off-Chip PT Buffer (128 Bytes)
0x010102A0 – 0x010107FF	General-Purpose RAM (1376 Bytes)

Contents	Address
XPT7 / SOF0 Linked List Start Add.	0x010100E0
XPT6 / SAM0 Linked List Start Add.	0x010100E4
XPT5 / SOF1 Linked List Start Add.	0x010100E8
XPT4 / SAM1 Linked List Start Add.	0x010100EC
XPT3 Linked List Start Add.	0x010100F0
XPT2 Linked List Start Add.	0x010100F4
XPT1 Linked List Start Add.	0x010100F8

Figure 20. MP Parameter RAM

MP interrupt vectors

Table 2 and Table 3 show the MP interrupts and traps and their vector addresses.

Table 2. Maskable Interrupts

IE BIT (TRAP#)	NAME	VECTOR ADDRESS	MASKABLE INTERRUPT
0	ie	0x01010180	
2	fi	0x01010188	Floating-point invalid
3	fz	0x0101018C	Floating-point divide-by-zero
5	fo	0x01010194	Floating-point overflow
6	fu	0x01010198	Floating-point underflow
7	fx	0x0101019C	Floating-point inexact
8	f0	0x010101A0	Frame timer 0
9	f1	0x010101A4	Frame timer 1
10	ti	0x010101A8	MP timer
11	x1	0x010101AC	External interrupt 1 ($\overline{EINT1}$)
12	x2	0x010101B0	External interrupt 2 ($\overline{EINT2}$)
14	mf	0x010101B8	Memory fault
15	io	0x010101BC	Integer overflow
16	p0	0x010101C0	PP0 message
17	p1	0x010101C4	PP1 message
18	p2	0x010101C8	PP2 message
19	p3	0x010101CC	PP3 message
25	mi	0x010101E4	MP message
26	pc	0x010101E8	Packet transfer complete
27	pb	0x010101EC	Packet transfer busy
28	bp	0x010101F0	Bad packet transfer
29	x3	0x010101F4	External interrupt 3 ($\overline{EINT3}$)
30	x4	0x010101F8	External interrupt 4 ($\overline{LINT4}$)
31	pe	0x010101FC	PP error

Table 3. Nonmaskable Traps

TRAP NO.	NAME	VECTOR ADDRESS	NONMASKABLE TRAP
32	e1	0x01010200	Emulator trap1 (reserved)
33	e2	0x01010204	Emulator trap2 (reserved)
34	e3	0x01010208	Emulator trap3 (reserved)
35	e4	0x0101020C	Emulator trap4 (reserved)
36	fe	0x01010210	Floating-point error
37		0x01010214	Reserved
38	er	0x01010218	Illegal MP instruction
39		0x0101021C	Reserved
72 to 415		0x010102A0 to 0x010107FC	System or user defined

MP opcode formats

The three basic classes of MP instruction opcodes are; short immediate, three register, and long immediate. Figure 21 shows the opcode structure for each class of instruction.

	31–27	26–22	21–15	14–0
Short Immediate	Dest	Source 2	Opcode	15-Bit Immediate

	31–27	26–22	21	20	19–13	12	11–5	4–0
Three Register	Dest	Source 2	1	1	Opcode	0	Options	Source 1

	31–27	26–22	21	20	19–13	12	11–5	4–0
Long immediate	Dest	Source 2	1	1	Opcode	1	Options	Source 1
	32-Bit Long Immediate							

Figure 21. MP Opcode Formats

MP opcode summary

Table 4 through Table 6 show the opcode formats for the MP. Table 7 summarizes the master processor instruction set.

Table 4. Short-Immediate Opcodes

	31–28	27	26–22	21	20	19	18	17	16	15	14–12	11	10	9–5	4–0
illop0	Dest		Source	0	0	0	0	0	0	0	Unsigned Immediate				
trap	– – – –	E	– – – – –	0	0	0	0	0	0	1	Unsigned Trap Number				
cmnd	– – – –	–	– – – – –	0	0	0	0	0	1	0	Unsigned Immediate				
rdcr	Dest		– – – – –	0	0	0	0	1	0	0	Unsigned Control Register Number				
swcr	Dest		Source	0	0	0	0	1	0	1	Unsigned Control Register Number				
brcr	– – – –	–	– – – – –	0	0	0	0	1	1	0	Unsigned Control Register Number				
shift.dz	Dest		Source	0	0	0	1	0	0	0	– – –	i	n	Endmask	Rotate
shift.dm	Dest		Source	0	0	0	1	0	0	1	– – –	i	n	Endmask	Rotate
shift.ds	Dest		Source	0	0	0	1	0	1	0	– – –	i	n	Endmask	Rotate
shift.ez	Dest		Source	0	0	0	1	0	1	1	– – –	i	n	Endmask	Rotate
shift.em	Dest		Source	0	0	0	1	1	0	0	– – –	i	n	Endmask	Rotate
shift.es	Dest		Source	0	0	0	1	1	0	1	– – –	i	n	Endmask	Rotate
shift.iz	Dest		Source	0	0	0	1	1	1	0	– – –	i	n	Endmask	Rotate
shift.im	Dest		Source	0	0	0	1	1	1	1	– – –	i	n	Endmask	Rotate
and.tt	Dest		Source2	0	0	1	0	0	0	1	Unsigned Immediate				
and.tf	Dest		Source2	0	0	1	0	0	1	0	Unsigned Immediate				
and.ft	Dest		Source2	0	0	1	0	1	0	0	Unsigned Immediate				
xor	Dest		Source2	0	0	1	0	1	1	0	Unsigned Immediate				
or.tt	Dest		Source2	0	0	1	0	1	1	1	Unsigned Immediate				
and.ff	Dest		Source2	0	0	1	1	0	0	0	Unsigned Immediate				
xnor	Dest		Source2	0	0	1	1	0	0	1	Unsigned Immediate				
or.tf	Dest		Source2	0	0	1	1	0	1	1	Unsigned Immediate				
or.ft	Dest		Source2	0	0	1	1	1	0	1	Unsigned Immediate				
or.ff	Dest		Source2	0	0	1	1	1	1	0	Unsigned Immediate				
ld	Dest		Base	0	1	0	0	M	SZ		Signed Offset				
ld.u	Dest		Base	0	1	0	1	M	SZ		Signed Offset				
st	Source		Base	0	1	1	0	M	SZ		Signed Offset				
dcache	– – – –	F	Source2	0	1	1	1	M	0	0	Signed Offset				
bsr	Link		– – – – –	1	0	0	0	0	0	A	Signed Offset				
jsr	Link		Base	1	0	0	0	1	0	A	Signed Offset				
bbz	BITNUM		Source	1	0	0	1	0	0	A	Signed Offset				
bbo	BITNUM		Source	1	0	0	1	0	1	A	Signed Offset				
bcnd	Cond		Source	1	0	0	1	1	0	A	Signed Offset				
cmp	Dest		Source2	1	0	1	0	0	0	0	Signed Immediate				
add	Dest		Source2	1	0	1	1	0	0	U	Signed Immediate				
sub	Dest		Source2	1	0	1	1	0	1	U	Signed Immediate				

– Reserved bit (code as 0)
A Annul delay slot instruction if branch taken
E Emulation trap bit
F Clear present flags
i Invert endmask
M Modify, write modified address back to register
n Rotate sense for shifting
SZ Size (0 = byte, 1 = halfword, 2 = word, 3 = doubleword)
U Unsigned form

MP opcode summary (continued)

Table 5. Long-Immediate and Three-Register Opcodes

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
trap	–	–	–	–	E	–	–	–	–	–	1	1	0	0	0	0	0	0	1	I	–	–	–	–	–	–	–	IND TR				
cmnd	–	–	–	–	–	–	–	–	–	–	1	1	0	0	0	0	0	1	0	I	–	–	–	–	–	–	–	Source1				
rdcr	Dest					–	–	–	–	–	1	1	0	0	0	0	1	0	0	I	–	–	–	–	–	–	–	IND CR				
swcr	Dest					Source					1	1	0	0	0	0	1	0	1	I	–	–	–	–	–	–	–	IND CR				
brcr	–	–	–	–	–	–	–	–	–	–	1	1	0	0	0	0	1	1	0	I	–	–	–	–	–	–	–	IND CR				
shift.dz	Dest					Source					1	1	0	0	0	1	0	0	0	0	i	n	Endmask					Rotate				
shift.dm	Dest					Source					1	1	0	0	0	1	0	0	1	0	i	n	Endmask					Rotate				
shift.ds	Dest					Source					1	1	0	0	0	1	0	1	0	0	i	n	Endmask					Rotate				
shift.ez	Dest					Source					1	1	0	0	0	1	0	1	1	0	i	n	Endmask					Rotate				
shift.em	Dest					Source					1	1	0	0	0	1	1	0	0	0	i	n	Endmask					Rotate				
shift.es	Dest					Source					1	1	0	0	0	1	1	0	1	0	i	n	Endmask					Rotate				
shift.iz	Dest					Source					1	1	0	0	0	1	1	1	0	0	i	n	Endmask					Rotate				
shift.im	Dest					Source					1	1	0	0	0	1	1	1	1	0	i	n	Endmask					Rotate				
and.tt	Dest					Source2					1	1	0	0	1	0	0	0	1	I	–	–	–	–	–	–	–	Source1				
and.tf	Dest					Source2					1	1	0	0	1	0	0	1	0	I	–	–	–	–	–	–	–	Source1				
and.ft	Dest					Source2					1	1	0	0	1	0	1	0	0	I	–	–	–	–	–	–	–	Source1				
xor	Dest					Source2					1	1	0	0	1	0	1	1	0	I	–	–	–	–	–	–	–	Source1				
or.tt	Dest					Source2					1	1	0	0	1	0	1	1	1	I	–	–	–	–	–	–	–	Source1				
and.ff	Dest					Source2					1	1	0	0	1	1	0	0	0	I	–	–	–	–	–	–	–	Source1				
xnor	Dest					Source2					1	1	0	0	1	1	0	0	1	I	–	–	–	–	–	–	–	Source1				
or.tf	Dest					Source2					1	1	0	0	1	1	0	1	1	I	–	–	–	–	–	–	–	Source1				
or.ft	Dest					Source2					1	1	0	0	1	1	1	0	1	I	–	–	–	–	–	–	–	Source1				
or.ff	Dest					Source2					1	1	0	0	1	1	1	1	0	I	–	–	–	–	–	–	–	Source1				
ld	Dest					Base					1	1	0	1	0	0	M	SZ		I	S	D	–	–	–	–	–	Offset				
ld.u	Dest					Base					1	1	0	1	0	1	M	SZ		I	S	D	–	–	–	–	–	Offset				
st	Source					Base					1	1	0	1	1	0	M	SZ		I	S	D	–	–	–	–	–	Offset				
dcache	–	–	–	–	F	Source2					1	1	0	1	1	1	M	0	0	I	0	0	–	–	–	–	–	Source1				
bsr	Link					–	–	–	–	–	1	1	1	0	0	0	0	0	A	I	–	–	–	–	–	–	–	Offset				
jsr	Link					Base					1	1	1	0	0	0	1	0	A	I	–	–	–	–	–	–	–	Offset				
bbz	BITNUM					Source					1	1	1	0	0	1	0	0	A	I	–	–	–	–	–	–	–	Target				
bbo	BITNUM					Source					1	1	1	0	0	1	0	1	A	I	–	–	–	–	–	–	–	Target				
bcnd	Cond					Source					1	1	1	0	0	1	1	0	A	I	–	–	–	–	–	–	–	Target				
cmp	Dest					Source2					1	1	1	0	1	0	0	0	0	I	–	–	–	–	–	–	–	Source1				
add	Dest					Source2					1	1	1	0	1	1	0	0	U	I	–	–	–	–	–	–	–	Source1				
sub	Dest					Source2					1	1	1	0	1	1	0	1	U	I	–	–	–	–	–	–	–	Source1				

– Reserved bit (code as 0)
D Direct external access bit
E Emulation trap bit
F Clear present flags
i Invert endmask
I Long immediate
M Modify, write modified address back to register
n Rotate sense for shifting
S Scale offset by data size
SZ Size (0 = byte, 1 = halfword, 2 = word, 3 = doubleword

MP opcode summary (continued)

Table 6. Miscellaneous Instruction Opcodes

	31–27	26–22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4–0
vadd	Mem Src/Dst	Source2/Dest	1	1	1	1	0	–	0	0	0	l	–	m	P	–	d	m	s	Source1
vsub	Mem Src/Dst	Source2/Dest	1	1	1	1	0	–	0	0	1	l	–	m	P	–	d	m	s	Source1
vmpy	Mem Src/Dst	Source2/Dest	1	1	1	1	0	–	0	1	0	l	–	m	P	–	d	m	s	Source1
vmsub	Mem Src/Dst	Dest	1	1	1	1	0	a	0	1	1	l	a	m	P	Z	–	m	–	Source1
vrnd(FP)	Mem Src/Dst	Dest	1	1	1	1	0	a	1	0	0	l	a	m	P	PD		m	s	Source1
vrnd(Int)	Mem Src/Dst	Dest	1	1	1	1	0	–	1	0	1	l	–	m	P	–	d	m	s	Source1
vmac	Mem Src/Dst	Source2	1	1	1	1	0	a	1	1	0	l	a	m	P	Z	–	m	–	Source1
vmsc	Mem Src/Dst	Source2	1	1	1	1	0	a	1	1	1	l	a	m	P	Z	–	m	–	Source1
fadd	Dest	Source2	1	1	1	1	1	0	0	0	0	l	–	PD		P2		P1		Source1
fsub	Dest	Source2	1	1	1	1	1	0	0	0	1	l	–	PD		P2		P1		Source1
fmpy	Dest	Source2	1	1	1	1	1	0	0	1	0	l	–	PD		P2		P1		Source1
fdiv	Dest	Source2	1	1	1	1	1	0	0	1	1	l	–	PD		P2		P1		Source1
frndx	Dest	– – – – –	1	1	1	1	1	0	1	0	0	l	–	PD		RM		P1		Source1
fcmp	Dest	Source2	1	1	1	1	1	0	1	0	1	l	–	–	–	P2		P1		Source1
fsqrt	Dest	– – – – –	1	1	1	1	1	0	1	1	1	l	–	PD		–	–	P1		Source1
lmo	Dest	Source	1	1	1	1	1	1	0	0	0	–	–	–	–	–	–	–	–	– – – – –
rmo	Dest	Source	1	1	1	1	1	1	0	0	1	–	–	–	–	–	–	–	–	– – – – –
estop	– – – – –	– – – – –	1	1	1	1	1	1	1	1	0	–	–	–	–	–	–	–	–	– – – – –
illopF	– – – – –	– – – – –	1	1	1	1	1	1	1	1	1	C	–	–	–	–	–	–	–	– – – – –

–	Reserved bit (code as 0)
a	Floating-point accumulator select
C	Constant operands rather than register
d	Destination precision for vector (0 = sp, 1 = dp)
l	Long immediate 32-bit data
m	Parallel memory operation specifier
Mem Src/Dst	Vector store or load source/dst register
Dest	Destination register
P	Dest precision for parallel load/store (0 = single, 1 = double)
P1	Precision of source1 operand
P2	Precision of source2 operand
PD	Precision of destination result
RM	Rounding Mode (0 = N, 1 = Z, 2 = P, 3 = M)
s	Scale offset by data size
Z	Use 0 rather than accumulator

MP opcode summary (continued)

Table 7. Summary of MP Opcodes

INSTRUCTION	DESCRIPTION	INSTRUCTION	DESCRIPTION
add	Signed integer add	or.ff	Bitwise OR with 1s complement
and.tt	Bitwise AND	or.ft	Bitwise OR with 1s complement
and.ff	Bitwise AND with 1s complement	or.tf	Bitwise OR with 1s complement
and.ft	Bitwise AND with 1s complement	rdcr	Read control register
and.tf	Bitwise AND with 1s complement	rmo	Rightmost one
bbo	Branch bit one	shift.dz	Shift, disable mask, zero extend
bbz	Branch bit zero	shift.dm	Shift, disable mask, merge
bcnd	Branch conditional	shift.ds	Shift, disable mask, sign extend
br	Branch always	shift.ez	Shift, enable mask, zero extend
brcr	Branch control register	shift.em	Shift, enable mask, merge
bsr	Branch and save return	shift.es	Shift, enable mask, sign extend
cmnd	Send command	shift.iz	Shift, invert mask, zero extend
cmp	Integer compare	shift.im	Shift, invert mask, merge
dcache	Flush data cache sub-block	st	Store register into memory
estop	Emulation stop	sub	Signed integer subtract
fadd	Floating-point add	swcr	Swap control register
fcmp	Floating-point compare	trap	Trap
fdiv	Floating-point divide	vadd	Vector floating-point add
fmpy	Floating-point multiply	vmac	Vector floating-point multiply and add to accumulator
frndx	Floating-point convert/round	vmpy	Vector floating-point multiply
fsqrt	Floating-point square root	vmsc	Vector floating-point multiply and subtract from accumulator
fsub	FLoating-point subtract	vmsub	Vector floating-point subtract accumulator from source
illop	Illegal operation	vrnd(FP)	Vector round with floating-point input
jsr	Jump and save return	vrnd(Int)	Vector round with integer input
ld	Load signed into register	vsub	Vector floating-point subtract
ld.u	Load unsigned into register	xnor	Bitwise exclusive NOR
lmo	Leftmost one	xor	Bitwise exclusive OR
or.tt	Bitwise OR		

PP architecture

The parallel processor (PP) is a 32-bit integer DSP optimized for imaging and graphics applications. Each PP can execute in parallel; a multiply, ALU operation, and two memory accesses within a single instruction. This internal parallelism allows a single PP to achieve over 500 million operations per second for certain algorithms. The PP has a three-input ALU that supports all 256 three input Boolean combinations and many combinations of arithmetic and Boolean functions. Data-merging and bit-to-byte, bit-to-word, and bit-to-halfword translations are supported by hardware in the input data path to the ALU. Typical tasks performed by a PP include:

- Pixel-intensive processing
  - Motion estimation
  - Convolution
  - PixBLTs
  - Warp
  - Histogram
  - Mean square error
- Domain transforms
  - DCT
  - FFT
  - Hough
- Core graphics functions
  - Line
  - Circle
  - Shaded fills
  - Fonts
- Image Analysis
  - Segmentation
  - Feature extraction
- Bit-stream encoding/decoding
  - Data merging
  - Table look-ups

PP functional block diagram

Figure 22 shows a block diagram of a parallel processor. Key features of the PP include:

- 64-bit instruction word (supports multiple parallel operations)
- Three-stage pipeline for fast instruction cycle
- Numerous registers
  - 8 data, 10 address, 6 index registers
  - 20 other user-visible registers
- Data Unit
  - 16x16 integer multiplier (optional dual 8x8)
  - Splittable 3-input ALU
  - 32-bit barrel rotator
  - Mask generator
  - Multiple-status flag expander for translations to/from 1 bit-per-pixel space.
  - Conditional assignment of data unit results
  - Conditional source selection
  - Special processing hardware
    Leftmost one / rightmost one
    Leftmost bit change / rightmost bit change
- Memory addressing
  - Two address units (global & local) provide up to two 32-bit accesses in parallel with data unit operation.
  - 12 addressing modes (immediate and indexed)
  - Byte, halfword, and word addressability
  - Scaled indexed addressing
  - Conditional assignment for loads
  - Conditional source selection for stores
- Program flow
  - Three hardware loop controllers
    Zero overhead looping / branching
    Nested loops
    Multiple loop endpoints
  - Instruction cache management
  - PC mapped to register file
  - Interrupts for messages and context switching
- Algebraic assembly language

PP functional block diagram (continued)

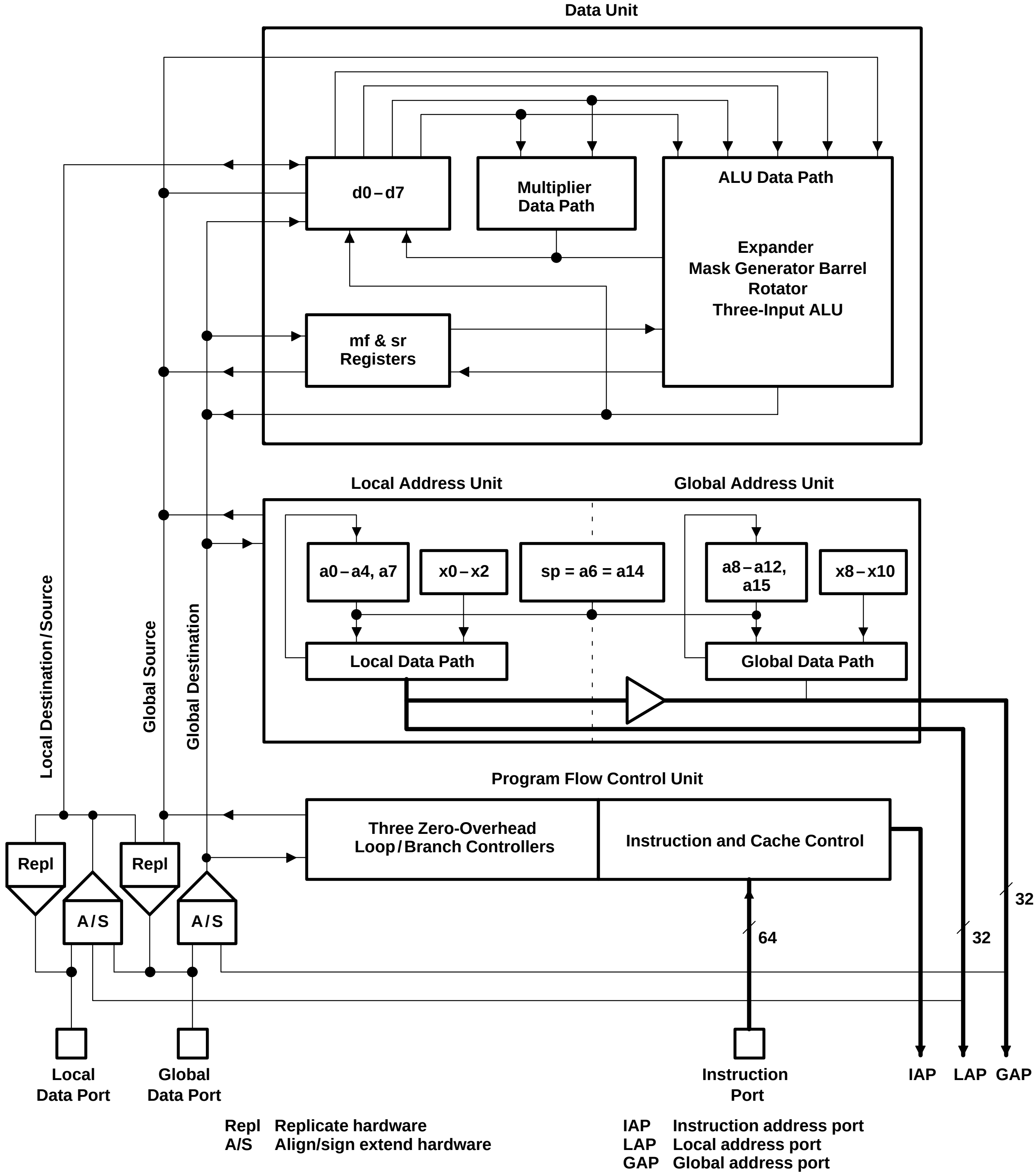


Figure 22. PP Block Diagram

PP registers

The PP contains many general-purpose registers, status registers, and configuration registers. All PP registers are 32-bit registers. Figure 23 shows the accessible registers of the PP blocks.

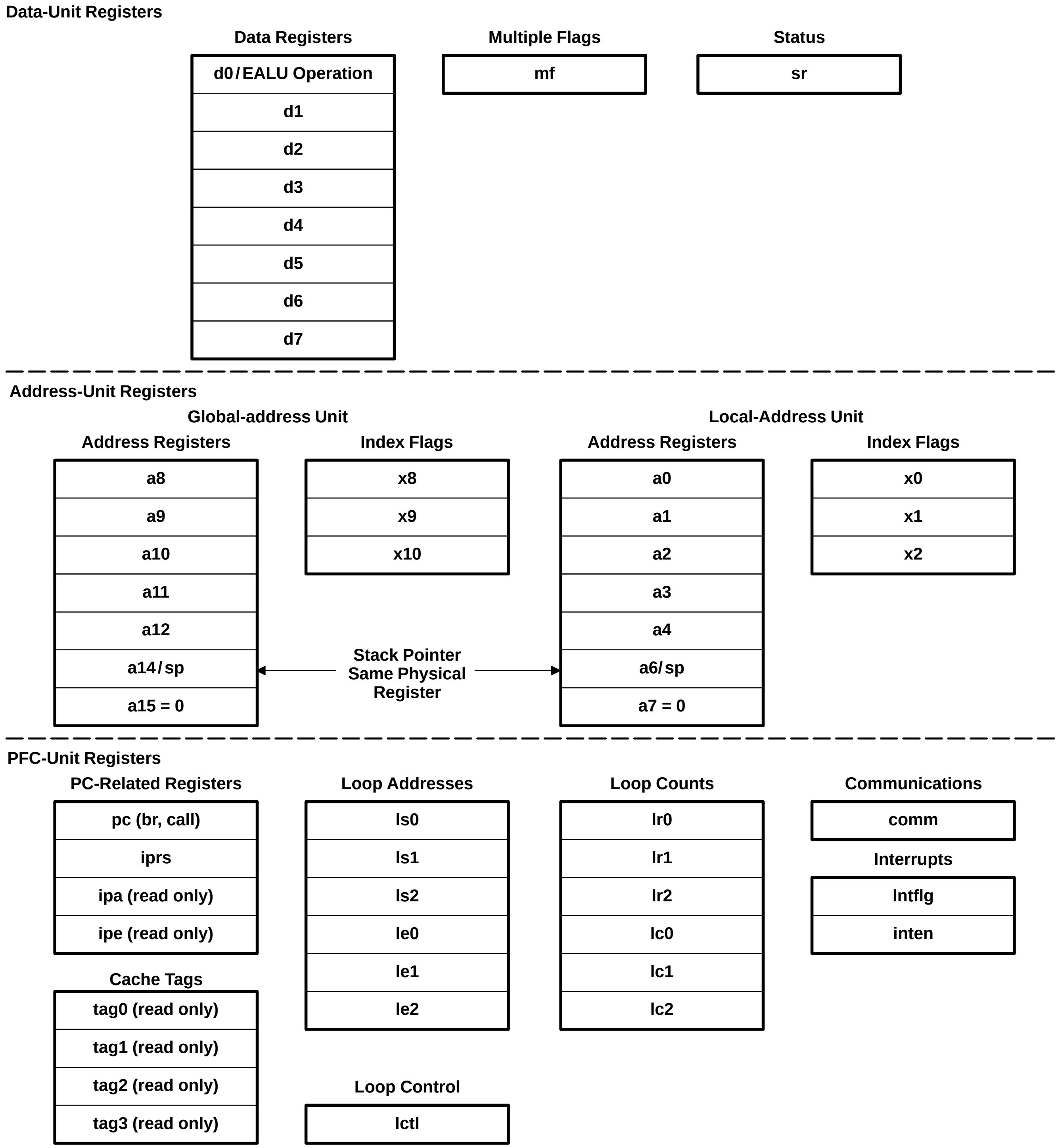


Figure 23. PP Registers

PP data-unit registers

The data unit contains eight 32-bit general-purpose data registers (d0–d7) referred to as the D registers. The d0 register also acts as the control register for EALU operations.

d0 register

Figure 24 shows the format when d0 is used as the EALU control register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMOD				A	EALU Function Code								C	I	S	N	E	F	–	–	–	DMS		M	R	–	DBR				

FMOD	Function modifiers	E	Explicit multiple carry-In
A	Arithmetic enable	F	Expanded mf
C	EALU carry-In	DMS	Default multiply shift amount
I	Invert-carry-In	M	Split multiply
S	Sign extend	R	Rounded multiply
N	Nonmultiple mask	DBR	Default barrel rotate amount

Figure 24. d0 Format for EALU Operations

multiple flags (mf) register

The mf register records status information from each split ALU segment for multiple arithmetic operations. The mf register may be expanded to generate a mask for the ALU. Figure 25 shows the mf register format.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 25. mf Register Format

status register (sr)

The sr contains status and control bits for the PP ALU. Figure 26 shows the sr register format.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N	C	V	Z	LV	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	MSS		R	Msize			Asize		

N	Negative status bit	MSS	mf Status selection
C	Carry status bit		00 – set by zero; 10 – set by extended result
V	Overflow status bit		01 – set by sign; 11 – reserved
Z	Zero status bit	Msize	Expander data size
LV	Latched overflow	Asize	Split ALU data size
R	Rotation bit		

Figure 26. sr Format

PP address-unit registers

address registers

The address unit contains ten 32-bit address registers which contain the base address for address computations or which can be used for general-purpose data. The registers a0 – a4 are used for local address computations and registers a8–a12 are used for global-address computations.

index registers

The six 32-bit index registers contain index values for use with the address registers in address computations or they can be used for general-purpose data. Registers x0 – x2 are used by the local-address unit and registers x8 – x10 are used by the global-address unit.

stack pointer (sp)

The sp contains the address of the bottom of the PP's system stack. The stack pointer is addressed as a6 by the local-address unit and as a14 by the global-address unit. Figure 27 shows the sp register format.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word-Aligned Address																														0	0

Figure 27. sp Register Format

zero register

The zero registers are read-as-zero address registers for the local address unit (a7) and global-address unit (a15). Writes to the registers are ignored and can be specified when operational results are to be discarded. Figure 28 shows the zero register format.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28. zero Registers

PP program flow control (PFC) unit registers

loop registers

The loop registers control three levels of zero-overhead loops. The 32-bit loop start registers (ls0 – ls2) and loop-end registers (le0 – le2) contain the starting and ending addresses for the loops. The loop-counter registers (lc0 – lc2) contain the number of repetitions remaining in their associated loops. The lr0 – lr2 registers are loop reload registers used to support nested loops. The format for the loop-control (lctl) register is shown in Figure 29. There are also six special write-only mappings of the loop-reload registers. The lrs0 – lrs2 codes are used for fast initialization of ls*n*, lr*n*, and lc*n* registers for multi-instruction loops while the lrse0 – lrse2 codes are used for single instruction-loop fast initialization.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	E	LCD2			E	LCD1			E	LCD0		
																				le2				le1				le0			

E Loop End Enable
LCDn Loop Counter Designator
000 – None 010 – lc1
001 – lc0 011 – lc2
1xx – reserved

Figure 29. lctl Register

pipeline registers

The PFC unit contains a pointer to each stage of the PP pipeline. The pc contains the program counter which points to the instruction being fetched. The ipa points to the instruction in the address stage of the pipeline and the ipe points to the instruction in the execute stage of the pipeline. The instruction pointer return-from-subroutine (iprs) register contains the return address for a subroutine call. Figure 30 shows the variable pipeline register format.

pipeline registers (continued)

pc

31–3	2	1	0
PC (29-Bit Doubleword Address)	–	G	L

G – Global Interrupt Enable   L – Loop Inhibit

↓

ipa

31–0
32-Bit Copy of the Previous pc Register Value

↓

ipe

31–0
32-Bit Copy of the Previous ipa Register Value

iprs

31–3	2	1	0
29-Bit Doubleword Return Address	–	–	–

Figure 30. Pipeline Registers

interrupt registers

The interrupt-enable (inten) register allows individual interrupts to be enabled and configures the interrupt flag (intflg) register operation. The intflg register contains the interrupt flag bits. Interrupt priority increases moving from left to right on intflg. Figure 31 shows the PP-interrupt register format.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
inten	r	r	r	r	E	E	E	E	–	–	–	E	E	E	E	–	–	E	–	–	–	–	–	–	–	–	–	–	–	–	–	W
					PP3MSG	PP2MSG	PP1MSG	PP0MSG				MPMSG	PTEND	PTERR	PTQ			TASK														
intflg	r	r	r	r	I	I	I	I	–	–	–	I	I	I	I	–	–	I	–	–	–	–	–	–	–	–	–	–	–	–	–	–

r Reserved (write as 0)
E Enable Interrupt
W Write Mode
  0 – writing 1 clears intflg
  1 – writing 1 sets intflg
PPnMSG PPn message Interrupt

MPMSG MP Message Interrupt
PTEND Packet Transfer Complete
PTERR Packet-Transfer Error
PTQ Packet Transfer Queued
TASK MP Task Interrupt

Figure 31. PP-Interrupt Registers

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77251–1443

communication (comm) register

The comm register contains the packet-transfer handshake bits and PP indicator bits. Figure 32 shows the comm register format.

31	30	29	28	27–17	16–13	12–9	8–7	6–3	2–0
H	S	Q	P	– – – – – – – – – – –	– – – –	– – – –	– –	– – – –	PP#
					MAREV	MIREV	DEV		

H	High-priority packet transfer	PP#	pp Number (read only)
S	Packet-transfer suspend		000 – pp0 010 – pp2
Q	Packet transfer queued		001 – pp1 011 – pp3
P	Submit packet transfer request		1xx – not implemented
MAREV	Major revision #		
MIREV	Minor revision #		
DEV Device	00 – 320C80		
	10 – 320C82		

Figure 32. comm Register

cache-tag registers

The tag0 – tag3 registers contain the tag address and sub-block present bits for each cache block. Figure 33 shows the cache tag registers.

31–9	8	7	6	5	4	3	2	1–0
23-Bit Tag Address	P	P	P	P	–	–	–	LRU
Sub-Block #	3	2	1	0				

P Present bit
LRU Least-recently-used code
00 – mru 10 – next lru
01 – next mru 11 – lru

Figure 33. Cache Tag Registers

PP cache architecture

Each PP has its own 2K-byte instruction cache. Each cache is divided into four blocks and each block is divided into four sub-blocks containing 16 64-bit instructions each. Cache misses cause one sub-block to be loaded into cache. Figure 34 shows the cache architecture for one of the four sets in each cache. Figure 35 shows how addresses map into the cache using the cache tags and address bits.

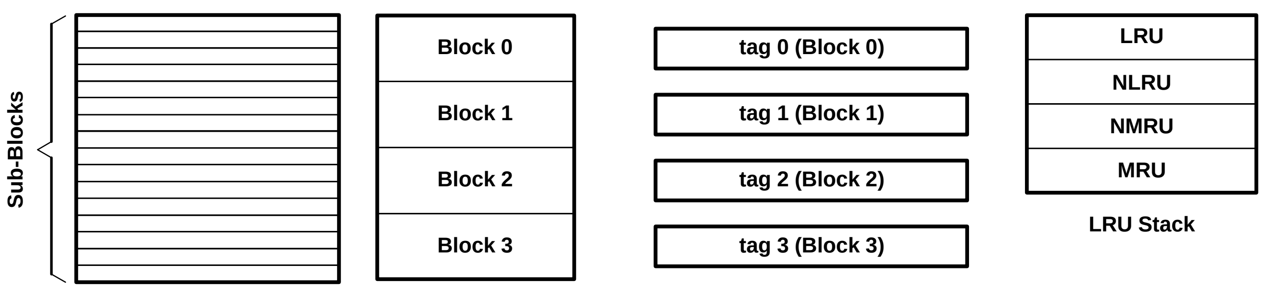


Figure 34. PP Cache Architecture

31–9	8–7	6–3	2–0
23-Bit Tag Value	sub	instruction	ignored

sub – sub-block

Figure 35. pc Register Cache-Address Mapping

PP parameter RAM

The parameter RAM is a, 2K-byte, on-chip RAM which contains PP-interrupt vectors, PP-requested TC task buffers, and a general-purpose area. The parameter RAM does not use the cache memory. Figure 36 shows the parameter RAM address map.

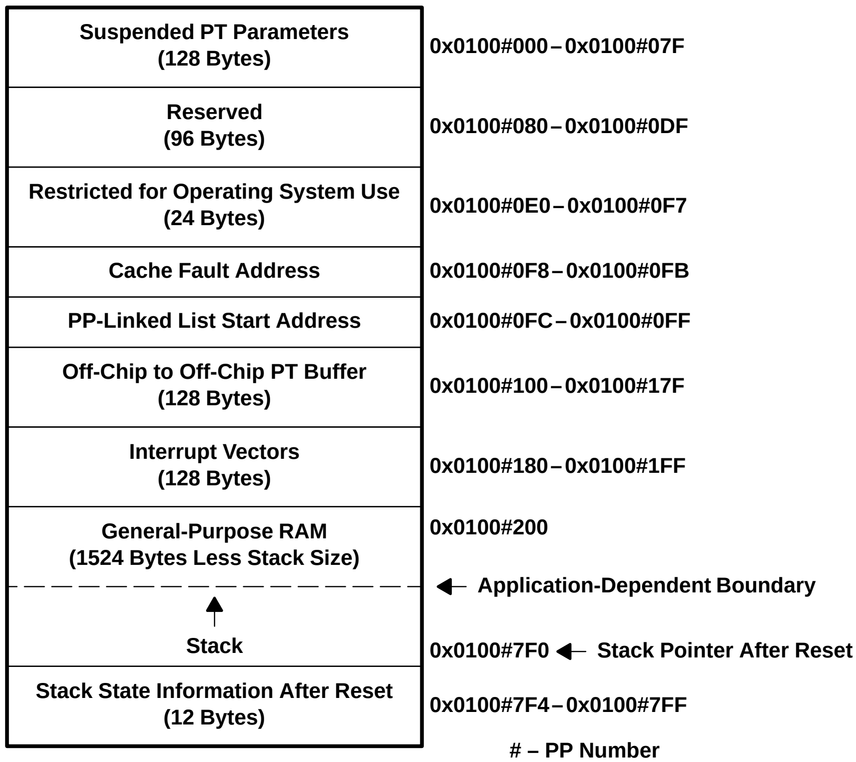


Figure 36. PP Parameter RAM

PP interrupt vectors

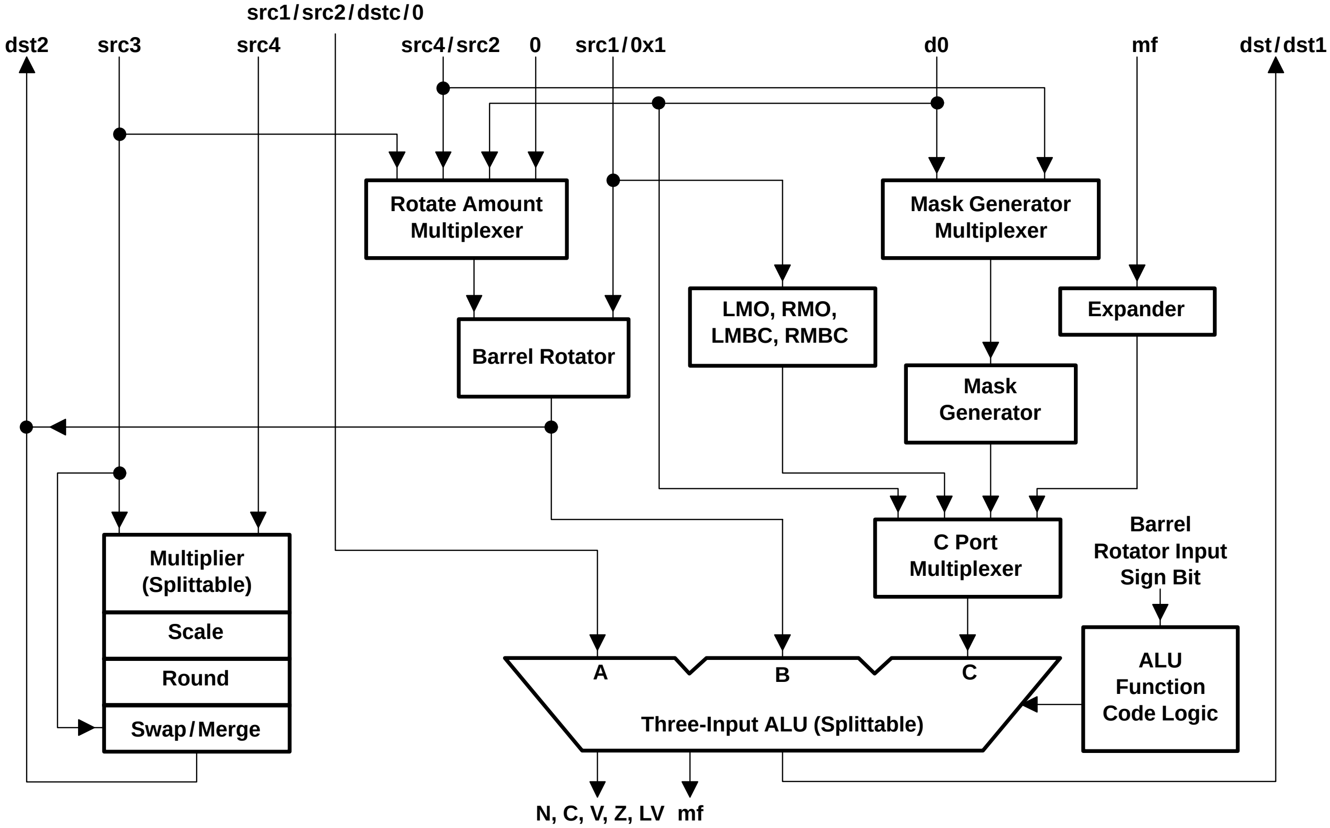
The PP interrupts and their vector addresses are shown in Table 8.

Table 8. PP-Interrupt Vectors

NAME	VECTOR ADDRESS	INTERRUPT
TASK	0x0100#1B8	Task Interrupt
PTQ	0x0100#1C4	Packet Transfer Queued
PTERR	0x0100#1C8	Packet-Transfer Error
PTEND	0x0100#1CC	Packet-Transfer End
MPMSG	0x0100#1D0	MP Message
PP0MSG	0x0100#1E0	PP0 Message
PP1MSG	0x0100#1E4	PP1 Message
PP2MSG	0x0100#1E8	PP2 Message
PP3MSG	0x0100#1EC	PP3 Message

PP data-unit architecture

The data unit has independent data paths for the ALU and the multiplier, each with its own set of hardware functions. The multiplier data path includes a 16 × 16 multiplier, a halfword swapper, and rounding hardware. The ALU data path includes a 32-bit three-input ALU, a barrel rotator, mask generator, mf expander, left/rightmost one and left/rightmost bit-change logic, and several multiplexers. Figure 37 shows the data-unit block diagram.



Legend:

src1	Any register, D reg only for l/rmo, l/rmbc hardware	dst2	D reg only
scr2	D reg or sometimes 15/32-bit immediate	dstc	D reg only (destination companion reg source)
scr3	D reg only	0x1	Constant
scr4	D reg only	0	Constant
dst/dst1	Any register	d0	5 LSBs of d0

Figure 37. Data Unit Block Diagram

The PP's ALU can be split into one 32-bit ALU, two 16-bit ALUs, or four 8-bit ALUs. Figure 38 shows the multiple arithmetic data flow for the case of a four 8-bit split of the ALU (called multiple-byte arithmetic). The ALU operates as independent parallel ALUs where each ALU receives the same function code.

PP data-unit architecture (continued)

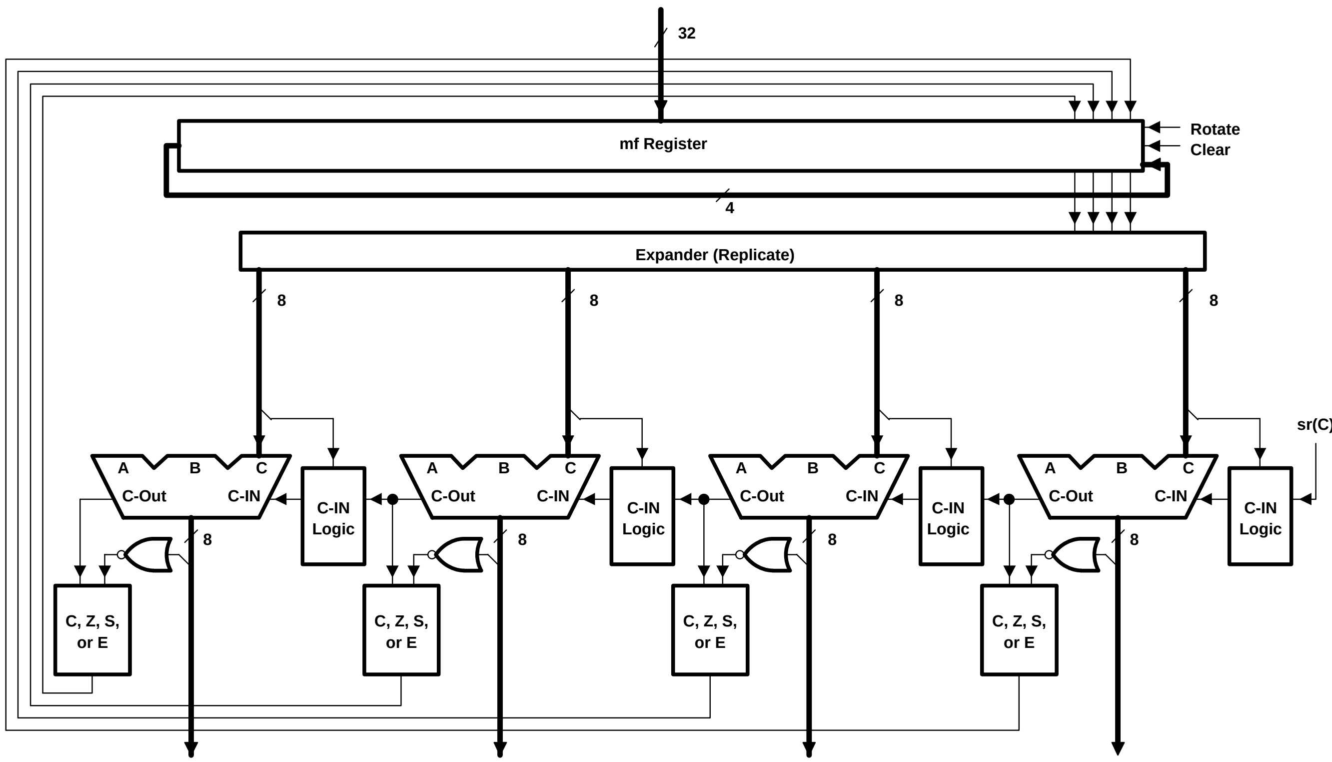


Figure 38. Multiple-Byte Arithmetic Data Flow

PP multiplier

The PP’s hardware multiplier can perform one 16x16 multiply with a 32-bit result or two 8x8 multiplies with two 16-bit results in a single cycle. A 16x16 multiply can use signed or unsigned operands as shown in Figure 39.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	S	Signed Input														

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	S	Signed × Signed Result																													

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Unsigned Input															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Unsigned × Unsigned Result																															

Figure 39. 16 x 16 Multiplier Data Formats

When performing two simultaneous 8x8 split multiplies, the first input word contains unsigned byte operands and the second input word contains signed or unsigned byte operands. These formats are shown in Figure 40 and Figure 41.

PP multiplier (continued)

31–16	15–8	7–0
X X X X X X X X X X X X X X X X	Unsigned Input 1b	Unsigned Input 1a

31–16	15	14–8	7	6–0
X X X X X X X X X X X X X X X X	S	Signed Input 2b	S	Signed Input 2a

31	30–16	15	14–0
S	1b × 2b Signed Result	S	1a × 2a Signed Result

Figure 40. Signed Split Multiply Data Formats

31–16	15–8	7–0
X X X X X X X X X X X X X X X X	Unsigned Input 1b	Unsigned Input 1a

31–16	15–8	7–0
X X X X X X X X X X X X X X X X	Unsigned Input 2b	Unsigned Input 2a

31–16	15–0
1b × 2b Unsigned Result	1a × 2a Unsigned Result

Figure 41. Unsigned Split Multiply Data Formats

PP program-flow-control unit architecture

The PP has a three-stage fetch, address, execute (FAE) pipeline as shown in Figure 42. The pc, ipa, and ipe registers point to the address of the instruction in each stage of the pipeline. On each cycle in which the pipeline advances, ipa is copied into ipe, pc is copied into ipa, and the pc is incremented by one instruction (8 bytes).

The program-flow-control (pfc) unit performs instruction fetching and decoding, loop control, and handshaking with the transfer controller. The pfc unit architecture is shown in Figure 43.

Instruction	T1	T2	T3	T4	T5
One	Fetch	Address	Execute		
Two		Fetch	Address	Execute	
Three			Fetch	Address	Execute

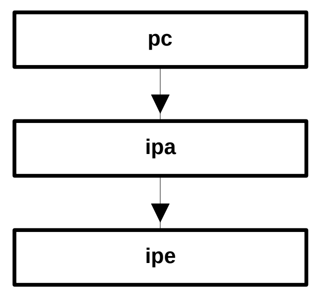


Figure 42. FAE-Instruction Pipeline

PP program-flow-control unit architecture (continued)

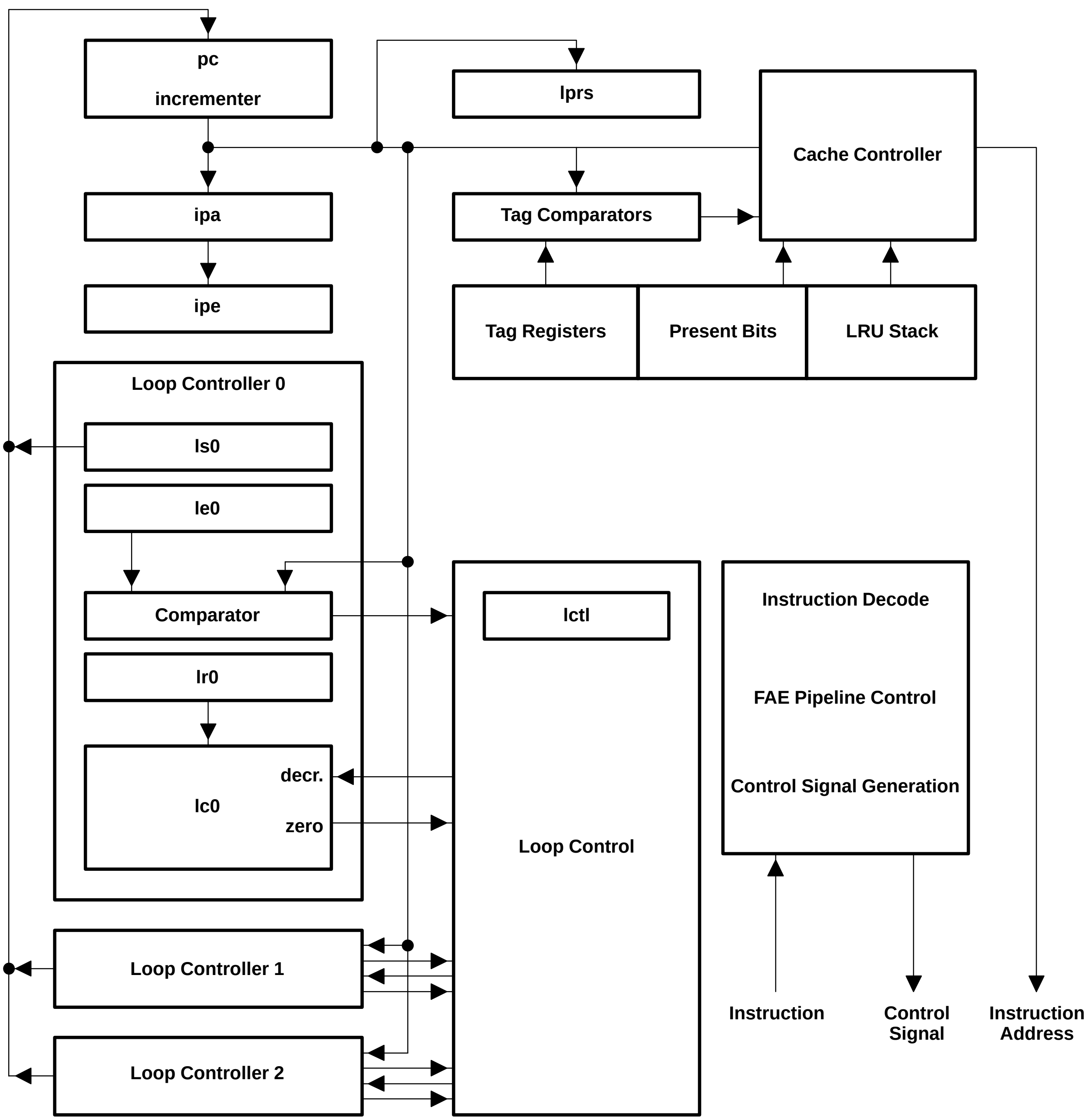


Figure 43. Program-Flow-Control Unit Block Diagram

PP address-unit architecture

The PP has both a local- and global-address unit which operate independently of each other. The address units support twelve different addressing modes. In place of performing a memory access, either or both of the address units can perform an address computation that is written directly to a PP register instead of being used for a memory access. This address-unit arithmetic provides additional arithmetic operation to supplement the data unit during compute-intensive algorithms. Figure 44 shows the address-out architecture.

PP address-unit architecture (continued)

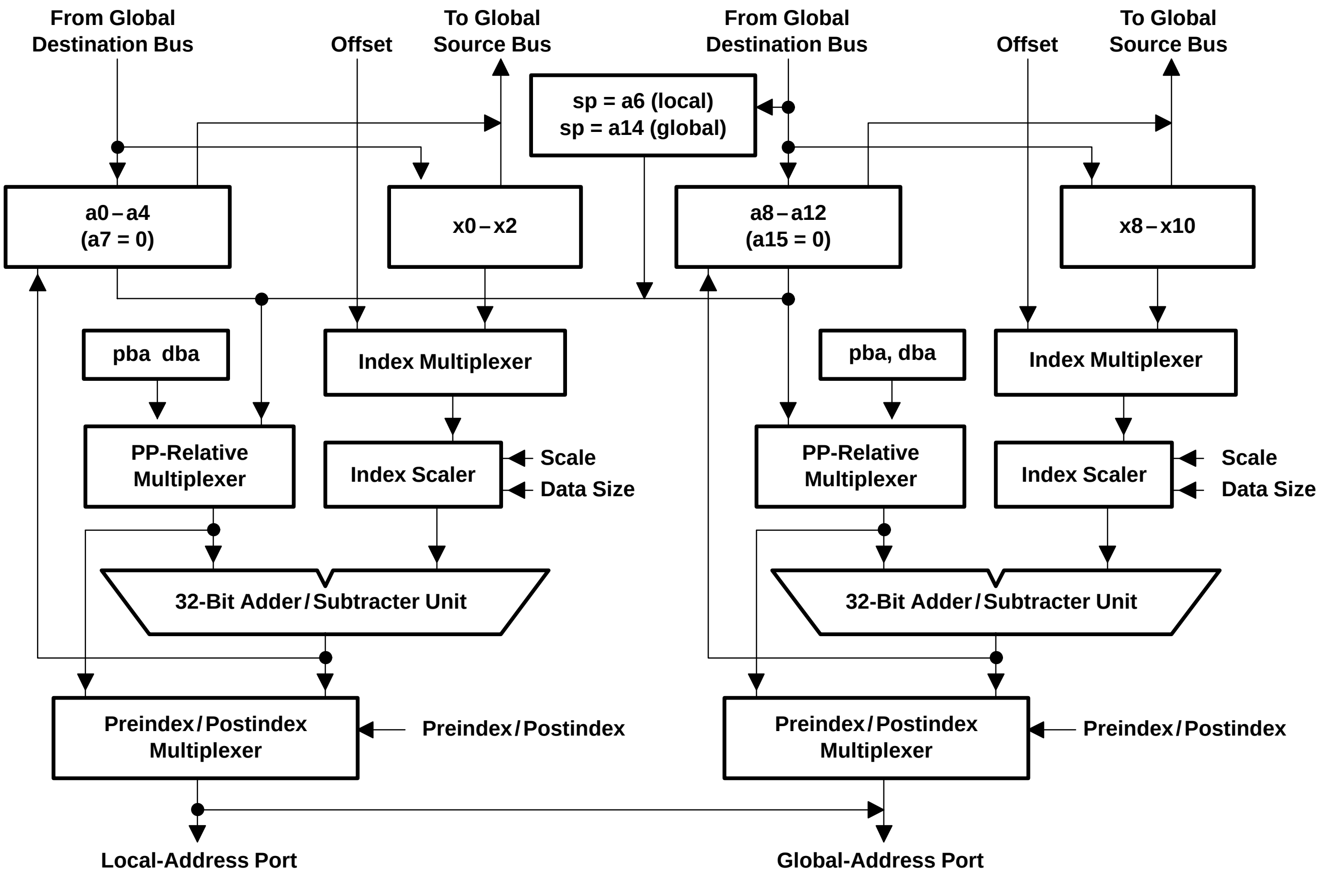


Figure 44. Address-Unit Architecture

PP instruction set

PP instructions are represented by algebraic expressions for the operations performed in parallel by the multiplier, ALU, global-address unit, and local-address unit. The expressions use the || symbol to indicate operations that are to be performed in parallel. The PP ALU operator syntax is shown in Table 9. The data unit operations (multiplier and ALU) are summarized in Table 10 and the parallel transfers (global and local) are summarized in Table 11.

PP instruction set (continued)

Table 9. PP Operators by Precedence

OPERATOR	FUNCTION
src1 [n] src1–1	Select odd (n=true) or even (n=false) register of D register pair based on negative condition code
( )	Subexpression delimiters
@mf	Expander operator
%	Mask generator
%%	Nonmultiple mask generator (EALU only)
%!	Modified mask generator (0xFFFFFFFF output for 0 input)
%%!	Nonmultiple shift right mask generator (EALU only)
\\	Rotate left
<<	Shift left (pseudo-op for rotate and mask)
>>u	Unsigned shift right
>> or >>s	Signed shift right
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
+	Addition
–	Subtraction
=[cond]	Conditional assignment
=[cond.pro]	Conditional assignment with status protection
=	Equate

PP instruction set (continued)

Table 10. Summary of Data-Unit Operations

<table>
<tr><td>Operation</td><td>Base set ALUs</td></tr>
<tr><td>Description</td><td>Perform an ALU operation specifying ALU function, 2 src and 1 dest operand, and operand routing. ALU function is one of 256 three-input Boolean operations or one of 17 arithmetic operations combined with one of 15 function modifiers.</td></tr>
<tr><td>Syntax</td><td>dst = [fmod] [[[cond [.pro]]]] ALU_EXPRESSION</td></tr>
<tr><td>Examples</td><td>d6 = (d6 ^ d4) & d2
d3 = [nn.nv] d1 –1</td></tr>
<tr><td>Operation</td><td>EALU || ROTATE</td></tr>
<tr><td>Description</td><td>Perform an extended ALU (EALU) operation (specified in d0) with one of two data routings to the ALU and optionally write the barrel rotator output to a second dest register. ALU Function is one of 256 Boolean or 256 arithmetic.</td></tr>
<tr><td>Syntax</td><td>dst1 = [[[cond [.pro]]]] ealu (src2, [dst2 =] [[[cond]] src1 [[[n]] src1–1] \\ src3, [%] src4)
dst1 = [fmod] [[[cond [.pro]]]] ealu (label:EALU_EXPRESSION [|| dst2 = [[cond]] src1 [[[n]] src1–1] \\ src3])</td></tr>
<tr><td>Examples</td><td>d7 = [nn] ealu(d2, d6 = [nn] d3\\d1, %d4)
d3 = mzc ealu(mylabel: d4 + (d5\\d6 & %d7) || d1 = d5\\d6)</td></tr>
<tr><td>Operation</td><td>MPY || ADD</td></tr>
<tr><td>Description</td><td>Perform a 16x16 multiply with optional parallel add or subtract. Condition code applies to both multiply and add.</td></tr>
<tr><td>Syntax</td><td>dst2 = [sign] [[[cond]]] src3 * src4 [|| dst = [[[cond[.pro]]]] src2 + src1 [[[n]] src1 –1]]
dst2 = [sign] [[[cond]]] src3 * src4 [|| dst = [[[cond[.pro]]]] src2 – src1 [[[n]] src1 –1]]</td></tr>
<tr><td>Example</td><td>d7 = u d6 * d5 || d5 = d4 – d1</td></tr>
<tr><td>Operation</td><td>MPY || SADD</td></tr>
<tr><td>Description</td><td>Perform a 16x16 multiply with a parallel right-shift and add or subtract. Condition code applies to multiply, shift, and add.</td></tr>
<tr><td>Syntax</td><td>dst2 = [sign] [[[cond]]] src3 * src4 || dst = [[[cond [.pro]]]] src2 + src1 [[[n]] src1 –1] >> –d0
dst2 = [sign] [[[cond]]] src3 * src4 || dst = [[[cond [.pro]]]] src2 – src1 [[[n]] src1 –1] >> –d0</td></tr>
<tr><td>Examples</td><td>d7 = u d6 * d5 || d5 = d4 – d1 >> –d0</td></tr>
<tr><td>Operation</td><td>MPY || EALU</td></tr>
<tr><td>Description</td><td>Perform a multiply and an optional parallel EALU. Multiply can use rounding, scaling, or splitting features.</td></tr>
<tr><td>Syntax</td><td>Generic Form:
dst2 = [sign] [[[cond]]] src3 * src4 || dst = [[[cond [.pro]]]] ealu[f] (src2, src1 [[[n]] src1 –1] \\ d0, %d0)
dst2 = [sign] [[[cond]]] src3 * src4 || ealu()
Explicit Form:
dst2 = [sign] [opt] [[[cond]]] src3 * src4 [<<dms] || dst1 = [fmod] [[[cond [.pro]]]] ealu (label: EALU_EXPRESSION)
dst2 = [sign] [opt] [[[cond]]] src3 * src4 [<<dms] || ealu (label)</td></tr>
<tr><td>Examples</td><td>d7 = [p] d5 * d3 || d2 = [p] ealu(d1, d6\\d0, %d0) ; generic form
d2 = m d4 * d7 || d3 = ealu (mylabel: d3 + d2 >> 9) ; explicit form</td></tr>
<tr><td>Operation</td><td>divi</td></tr>
<tr><td>Description</td><td>Perform one iteration of unsigned divide algorithm. Generates one quotient bit per execution using iterative subtraction.</td></tr>
<tr><td>Syntax</td><td>dst1 = [[[cond [.pro]]]] divi (src2, dst2 = [[cond]] src1 [[[n]] src1 –1])</td></tr>
<tr><td>Examples</td><td>d3 = divi (d1, d2 = d2)
d3 = divi (d1, d2 = d3[n]d2)</td></tr>
</table>

Legend:

[ ]	Optional parameter extension	cond	Condition code
[[ ]]	Square brackets ([ ]) must be used	fmod	Function modifier
pro	Protect status bits	dms	Default multiply shift amount
f	Use 1s compliment of d0	sign	u = unsigned, s = signed

PP instruction set (continued)

Table 10. Summary of Data-Unit Operations (Continued)

Misc. Operations	dint; eint; nop; dloop; eloop; qwait
Description	Globally disable interrupts; globally enable interrupts; do nothing in the data unit; globally enable looping; globally disable looping; wait until comm register Q bit is zero.

Syntax	`dint dloop`
	`eint eloop`
	`nop  qwait`

Legend:

[ ]	Optional parameter extension	cond	Condition code
[[ ]]	Square brackets ([ ]) must be used	fmod	Function modifier
pro	Protect status bits	dms	Default multiply shift amount
f	Use 1s compliment of d0	sign	u = unsigned, s = signed

PP instruction set (continued)

Table 11. Summary of Parallel Transfers

Operation	Load
Description	Transfer from memory into PP register
Syntax	dst = [sign] [size] [[[cond]]]* addrexp dst = [sign] [size] [[[cond]]]* an.element
Examples	`d3 = uh[n]* (a9++=[2])` `d1 = * a2.sMY_ELEMENT`
Operation	Store
Description	Transfer from PP register into memory
Syntax	* addrexp = [size] src [[[n]] src–1] * an.element = [size] src [[[n]] src–1]
Examples	`*––a2 = d3` `*a9.sMY_ELEMENT = a3`
Operation	Address unit arithmetic
Description	Compute address and store in PP register
Syntax	dst = [size] [[[cond]]] & * addrexp dst = [size] [[[cond]]] & * an.element
Examples	`d2 = &*(a3 + x0)` `a1 = &*a9.sMY_ELEMENT`
Operation	Move
Description	Transfer from PP register to PP register
Syntax	dst = [g] [[[cond]]] src
Examples	`x2 = mf` `d1 = g d3`
Operation	Field extract move
Description	Transfer from PP register to PP register extracting and right-aligning one byte or halfword
Syntax	dst = [sign] [size item]
Example	`d3 = ub2 d1`
Operation	Field replicate move
Description	Transfer from PP register to PP register replicating the LSbyte or LShalfword to 32 bits
Syntax	dst = r [size] [[cond]] src
Example	`d7 = rh d3`

Legend:

[ ]	Optional parameter extension	cond	Condition code
[[ ]]	Square brackets ([ ]) must be used	sign	u = unsigned, s = signed
g	Use global unit	size	b = byte, h = halfword, w = word (default)
item	0 = byte0/halfword0, 1 = byte1/halfword1, 2 = byte2, 3 = byte3		

PP opcode formats

A PP instruction uses a 64-bit opcode. The opcode is divided essentially into a data unit portion and a parallel transfer portion. There are five data unit opcode formats comprising bits 38–63 of the opcode. Bits 0–38 of the opcode specify one of 10 parallel transfer formats. An alphabetical list of the mnemonics used in Figure 45 for the data unit and parallel transfer portions of the opcode are shown in Table 12 and Table 13, respectively.

Data Unit Formats

Bits: 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 … 3 2 1 0

Format																	
A. Six-Operand (MPYIIADD, etc.)	0	1	1	oper	src3	dst2	dst1	src1	src4	src2	Parallel Transfers						
B. Base Set ALU (5-Bit Immediate)	1	class	A	ALU Operation	dst	src1	0	imm. src2	Parallel Transfers								
C. Base Set ALU (Register src2)	1	class	A	ALU Operation	dst	src1	1 0	–	src2	Parallel Transfers							
D. Base Set ALU (32-Bit Immediate)	1	class	A	ALU Operation	dst	src1	1 1	dstbank	s1bnk	cond	32-Bit Immediate						
E. Miscellaneous	1	0 0 0	1	–	0	–	0	–	0	–	0	– – – – – – –	0	Operation	Parallel Transfers		
	0 0	Reserved															
	0 1 0	Reserved															

Transfer Formats

Bits: 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Format																			
1. Double Parallel	Lmode	d	e	size	s	La	Gim/X	L	0bank	L	Gmode	reg	e	size	s	Ga	Lim/X		
2. Move II Local	Lmode	d	e	size	s	La	0	Lrm	dstbank	L	0 0	0 0	src	srcbank	dst	Lim/X			
3. Field Move II Local	Lmode	d	e	size	s	La	0	Lrm	dstbank	L	0 0	0 1	src	e	size	D	dst	Lim/X	
4. Local (Long Offset)	Lmode	reg	e	size	s	La	1	Lrm	bank	L	0 0	Local Long Offset / X							
5. Global (Long Offset)	0 0	Global Long Offset /X	bank	L	Gmode	reg	e	size	s	Ga	0	Grm							
6. Non-D DU II Local	Lmode	d	e	size	s	La	0	Lrm	Adstbank	L	0 0	1	– – – –	As1bank	– – –	Lim/X			
7. Conditional DU II Conditional Mode	0 0	–	cond	c	r	g	N C V Z	0	– –	dstbank	–	0 0	0 0	src	srcbank	dst	– – –		
8. Conditional DU II Conditional Field Move	0 0	–	cond	c	r	g	N C V Z	0	itm	dstbank	–	0 0	0 1	src	e	size	D	dst	– – –
9. Conditional DU II Conditional Global	0 0	–	cond	c	r	g	N C V Z	Gim/X	bank	L	Gmode	reg	e	size	s	Ga	1	Grm	
10. Conditional Non-D DU	0 0	–	cond	c	r	–	N C V Z	0	– –	Adstbank	–	0 0	1	– – – –	As1bank	– – – – – –			

Figure 45. PP Opcode Formats

PP opcode formats (continued)

Table 12. Data Unit Mnemonics

MNEMONIC	FUNCTION
A	A = 1 selects arithmetic operations, A = 0 selects Boolean operations
ALU Operation	For Boolean operation (A = 0) select the 8 ALU function signals. For arithmetic operation (A = 1), odd bits specify the ALU function and even bits define the ALU function modifiers.
class	Operation class, determines routing of ALU operands
cond	condition code
dst	D register destination or lower 3 bits of non-D register code
dst1	ALU dest. for MPY||ADD, MPY||EALU, or EALU||ROTATE operation. D register or lower 3 bits of non-D register code
dst2	Multiply dest. for MPY||ADD or MPY||EALU operation or rotate dest. for EALU||ROTATE operation. D register
dstbank	ALU register bank
imm.src2	5-bit immediate for src2 of ALU operation
32-Bit Immediate	32-bit immediate for src2 of ALU operation
oper	Six-operand data unit operation (MPY||ADD, MPY||SADD, MPY||EALU, EALU||ROTATE, divi)
Operation	Miscellaneous operation
src1	ALU source 1 register code (D register unless srcbank or s1bank is used)
src2	D register used as ALU source 2
src3	D register for multiplier source (MPY||ADD or MPY||EALU) or rotate amount (EALU||ROTATE)
src4	D reg for ALU C port operand or EALU||ROTATE mask generator input or multiplier source 2 for MPY||ADD, MPY||EALU
s1bnk	Bits 5-3 of src1 register code (bit 6 assumed to be 0)

PP opcode formats (continued)

Table 13. Parallel Transfer Mnemonics

MNEMONIC	FUNCTION
0bank	Bits 5–3 of global transfer source/destination register code (bit 6 assumed to be 0)
Adstbnk	Bits 6–3 of ALU destination register code
As1bank	Bits 6–3 of ALU source 1 register code
bank	Bits 6–3 of global (or local) store source or load destination
c	Conditional choice of D register for src1 operand of the ALU
C	Protect status register's carry bit
cond	Condition code
d	D register or lower 3 bits of register code for local transfer source/destination
D	Duplicate least significant data during moves
dst	The three lowest bits of the register code for move or field move destination
dstbank	Bits 6–3 of move destination register code
e	Sign extend local (bit 31), sign extend global (bit 9)
g	Conditional global transfer
Ga	Global address register for load, store, or address unit arithmetic
Gim / X	Global address unit immediate offset or index register
Gmode	Global unit addressing mode
Grm	Global PP-relative addressing mode
itm	Number of item selected for field extract move
L	L = 1 selects load operation, L = 0 selects store / address unit arithmetic operation
La	Local address register for load, store, or address unit arithmetic
Lim / X	Local address unit immediate offset or index register
Lmode	Local unit addressing mode
Lrm	Local PP-relative addressing mode
N	Protect status register's negative bit
r	Conditional write of ALU result
reg	Register number used with bank or 0bank for global load, store, or address unit arithmetic
s	Enable index scaling. Additional index bit for byte accesses or arithmetic operations (bit 28, local; bit 6, global)
size	Size of data transfer (bits 30 – 29, local; bits 8 – 7, global)
src	Three lowest bits of register code for register-register move source or non-field moves. D register source for field move
srcbank	Bits 6–3 of register code for register-register move source
V	Protect status register's overflow bit
Z	Protect status register's zero bit
–	Unused bit (fill with 0)

PP opcode formats (continued)

Table 14 summarizes the supported parallel-transfer formats, their formats, and whether the transfers are local or global. It also lists the allowed ALU operations and states whether conditions and status protection are supported.

Table 14. Parallel-Transfer Format Summary

FORMAT	ALU OPERANDS		Cond	Status Protection	GLOBAL TRANSFER				LOCAL TRANSFER			
					Move	Load / Store / AUA			Load / Store / AUA			
	dst1	src1			src → dst	s/d	Index	Rel	s/d	Index	Rel	Port
Double parallel	D	D	No	No	—	Lower	X/short	No	D	X/short	No	Local
Move || Local	D	D	No	No	Any→Any	—	—	—	D	X/short	Yes	Local
Field move || Local	D	D	No	No	D→Any	—	—	—	D	X/short	No	Local
Global (long offset)	D	D	No	No	—	Any	X/long	Yes	—	—	—	—
Local (long offset)	D	D	No	No	—	—	—	—	Any	X/long	Yes	Global
Non-D DU || Local	Any	Any	No	No	—	—	—	—	D	X/short	Yes	Global
Conditional move	D	D	Yes	Yes	Any→Any	—	—	—	—	—	—	—
Conditional field move	D	D	Yes	Yes	D→Any	—	—	—	—	—	—	—
Conditional global	D	D	Yes	Yes	—	Any	X/short	Yes	—	—	—	—
Conditional non-D DU	Any	Any	Yes	Yes	—	—	—	—	—	—	—	—
32-bit imm. base ALU	Any	Lower	Yes	No	—							

Legend:
- DU Data unit
- AUA Address unit arithmetic
- s/d Source/destination register
- Rel Relative addressing support

PP opcode formats (continued)

Table 15 shows the encoding used in the opcodes to specify particular PP registers. A 3-bit register field contains the three LSBs. The register codes are used for the src, src1, src2, src3, src4, dst, dst1, dst2, d, reg, Ga, La, Gim/X, and Lim/X opcode fields. The four MSBs specify the register bank which is concatenated to the register field for the full 7-bit code. The register bank codes are used for the dstbank, s1bnk, srcbank, 0bank, bank, Adstbnk, and As1bank opcode fields. When no associated bank is specified for a register field in the opcode, the D register bank is assumed. When the MSB of the bank code is not specified in the opcode (as in 0bank and s1bnk) it is assumed to be 0, indicating a lower register.

Table 15. PP Register Codes

LOWER REGISTERS (MSB OF BANK = 0)						UPPER REGISTERS (MSB OF BANK = 1)					
CODING		REGISTER	CODING		REGISTER	CODING		REGISTER	CODING		REGISTER
BANK	REG		BANK	REG		BANK	REG		BANK	REG	
0000	000	a0	0100	000	d0	1000	000	reserved	1100	000	lc0
0000	001	a1	0100	001	d1	1000	001	reserved	1100	001	lc1
0000	010	a2	0100	010	d2	1000	010	reserved	1100	010	lc2
0000	011	a3	0100	011	d3	1000	011	reserved	1100	011	reserved
0000	100	a4	0100	100	d4	1000	100	reserved	1100	100	lr0
0000	101	reserved	0100	101	d5	1000	101	reserved	1100	101	lr1
0000	110	a6 (sp)	0100	110	d6	1000	110	reserved	1100	110	lr2
0000	111	a7 (zero)	0100	111	d7	1000	111	reserved	1100	111	reserved
0001	000	a8	0101	000	reserved	1001	000	reserved	1101	000	lrse0
0001	001	a9	0101	001	sr	1001	001	reserved	1101	001	lrse1
0001	010	a10	0101	010	mf	1001	010	reserved	1101	010	lrse2
0001	011	a11	0101	011	reserved	1001	011	reserved	1101	011	reserved
0001	100	a12	0101	100	reserved	1001	100	reserved	1101	100	lrs0
0001	101	reserved	0101	101	reserved	1001	101	reserved	1101	101	lrs1
0001	110	a14 (sp)	0101	110	reserved	1001	110	reserved	1101	110	lrs2
0001	111	a15 (zero)	0101	111	reserved	1001	111	reserved	1101	111	reserved
0010	000	x0	0110	000	reserved	1010	000	reserved	1110	000	ls0
0010	001	x1	0110	001	reserved	1010	001	reserved	1110	001	ls1
0010	010	x2	0110	010	reserved	1010	010	reserved	1110	010	ls2
0010	011	reserved	0110	011	reserved	1010	011	reserved	1110	011	reserved
0010	100	reserved	0110	100	reserved	1010	100	reserved	1110	100	le0
0010	101	reserved	0110	101	reserved	1010	101	reserved	1110	101	le1
0010	110	reserved	0110	110	reserved	1010	110	reserved	1110	110	le2
0010	111	reserved	0110	111	reserved	1010	111	reserved	1110	111	reserved
0011	000	x8	0111	000	pc/call	1011	000	reserved	1111	000	reserved
0011	001	x9	0111	001	ipa/br	1011	001	reserved	1111	001	reserved
0011	010	x10	0111	010	ipe #	1011	010	reserved	1111	010	reserved
0011	011	reserved	0111	011	iprs	1011	011	reserved	1111	011	reserved
0011	100	reserved	0111	100	inten	1011	100	reserved	1111	100	tag0 #
0011	101	reserved	0111	101	intflg	1011	101	reserved	1111	101	tag1 #
0011	110	reserved	0111	110	comm	1011	110	reserved	1111	110	tag2 #
0011	111	reserved	0111	111	lctl	1011	111	reserved	1111	111	tag3 #

Read only

data unit operation code

For data unit opcode format A, a 4-bit operation code specifies one of 16 six-operand operations and an associated data path, as shown in Table 16.

Table 16. Six Operand Format Operation Codes

oper FIELD BIT				
60	**59**	**58**	**57**	**OPERATION TYPE**
0	u	0	s	MPY || ADD
0	u	1	f	MPYU || EALU
1	0	f	k	EALU || ROTATE
1	0	1	0	divi
1	1	u	s	MPY || SADD

Legend:
u Unsigned
f 1s complement EALU function code
s Subtract
k Use mask or mf expander

operation class code

The base set ALU opcodes (formats B, C, D) use an operation class code to specify one of eight different routings to the A, B, and C ports of the ALU, as shown in Table 17.

Table 17. Base Set ALU Class Summary

CLASS	DESTINATION	A PORT	B PORT	C PORT
000	dst	src2	src1	@mf
001	dst	dstc	src1 \ \ d0	src2
010	dst	dstc	src1	%src2
011	dst	dstc	src1 \ \ src2	%src2
100	dst	src2	src1 \ \ d0	%d0
101	dst	src2	src1 \ \ d0	@mf
110	dst	dstc	src1	src2
111	dst	src1	1 \ \ src2	src2

Legend:
\ \ Rotate left
@mf Expand function
% Mask generation
dstc Companion D reg
dst Destination Dreg or any reg if dstbank or Adstbnk is used with destination.
src2 Source D reg or immediate
srd1 Source D reg or any if As1bank is used or any lower reg if s1bnk is used

ALU-operation code

For base-set ALU Boolean opcodes (A=0), the ALU function is formed by a sum of Boolean products selected by the ALU operation opcode bits as shown in Table 18. For base-set arithmetic opcodes (A=1), the four odd ALU operation bits specify an arithmetic operation as described in Table 19 while the four even bits specify one of the ALU function modifiers as shown in Table 20.

Table 18. Base-Set ALU Boolean Function Codes

OPCODE BIT	PRODUCT TERM
58	A & B & C
57	~A & B & C
56	A & ~B & C
55	~A & ~B & C
54	A & B & ~C
53	~A & B & ~C
52	A & ~B & ~C
51	~A & ~B & ~C

Table 19. Base-Set Arithmetics

OPCODE BITS				CARRY IN	ALGEBRAIC DESCRIPTION	NATURAL FUNCTION	MODIFIED FUNCTION (IF DIFFERENT FROM NATURAL FUNCTION)
57	55	53	51				
0	0	0	0	x			
0	0	0	1	1	A – (B | C)	A – B <1<	
0	0	1	0	0	A + (B & ~C)	A + B <0<	
0	0	1	1	1	A – C	A – C	
0	1	0	0	1	A – (B | ~C)	A – B >1>	(A – (B & C)) if sign=0
0	1	0	1	1	A – B	A – B	
0	1	1	0	C(n)	A – (B & @mf | –B & ~@mf)	A + B / A – B	if class 0 or 5
				1/0	A + |B|	A + B / A – B	if class 1–4 or 6–7, A–B if sign=1
0	1	1	1	1	A – (B & C)	A – B>0>	
1	0	0	0	0	A + (B & C)	A + B>0>	
1	0	0	1	~C(n)	A + (B & @mf | –B & ~@mf)	A – B / A + B	if class 0 or 5
				0/1	A – |B|	A – B / A + B	if class 1–4 or 6–7, A+B if sign=1
1	0	1	0	0	A + B	A + B	
1	0	1	1	0	A + (B | ~C)	A + B >1>	(A + (B & C)) if sign=0
1	1	0	0	0	A + C	A + C	
1	1	0	1	1	A – (B & ~C)	A – B <0<	
1	1	1	0	0	A + (B | C)	A + B <1<	
1	1	1	1	0	(A & C) + (B & C)	field A + B	

Legend:
- C(n) LSB of each part of C port register
- >0> Zero-extend shift right
- <0< Zero-extend shift left
- >1> One-extend shift right
- <1< One-extend shift left

ALU-operation code (continued)

Table 20. Function Modifier Codes

FUNCTION MODIFIER BITS 58	56	54	52	MODIFICATION PERFORMED
0	0	0	0	Normal operation
0	0	0	1	cin
0	0	1	0	%! if maskgen instruction, lmo if not maskgen
0	0	1	1	%! and cin if maskgen instruction, rmo if not maskgen
0	1	0	0	A port = 0
0	1	0	1	A port = 0 and cin
0	1	1	0	A port = 0 and %! if maskgen, lmbc if not maskgen
0	1	1	1	A port = 0, %! and cin if maskgen, rmbc if not maskgen
1	0	0	0	mf bit(s) set by carry out(s). (mc)
1	0	0	1	mf bit(s) set based on status register MSS field. (me)
1	0	1	0	Rotate mf by Asize, mf bit(s) set by carry out(s). (mrc)
1	0	1	1	Rotate mf by Asize, mf bit(s) set based on status register MSS field. (mre)
1	1	0	0	Clear mf, mf bit(s) set by carry out(s). (mzc)
1	1	0	1	Clear mf, mf bit(s) set based on status register MSS field. (mze)
1	1	1	0	No setting of bits in mf register. (mx)
1	1	1	1	Reserved

Legend:
cin Carry in from sr(C)
lmbc Leftmost-bit change
lmo Leftmost one
%! Modified mask generator
rmbc Rightmost-bit change
rmo Rightmost one

miscellaneous operation code

For data-unit opcode format E, the operation field selects one of the miscellaneous operations codes as shown in Table 21.

Table 21. Miscellaneous Operation Codes

OPCODE BITS 43	42	41	40	39	MNEMONIC	OPERATION
0	0	0	0	0	nop	No data-unit operation. Status not modified
0	0	0	0	1	qwait	Wait until comm Q bit is clear
0	0	0	1	0	eint	Global-interrupt enable
0	0	0	1	1	dint	Global-interrupt disable
0	0	1	0	0	eloop	Global loop enable
0	0	1	0	1	dloop	Global loop disable
0	0	1	1	x	reserved	
0	1	x	x	x	reserved	
1	x	x	x	x	reserved	

addressing-mode codes

The Lmode (bits 35–38) and Gmode (bits 13–16) of the opcode specify the local and global transfer for various parallel transfer opcode formats (Lmode in formats 1, 2, 3, 4, and 6 and Gmode in formats 1, 5, and 9). Table 22 shows the coding for the addressing-mode fields.

Table 22. Addressing-Mode Codes

CODING	EXPRESSION	DESCRIPTION
00xx		Nop (nonaddressing mode operation)
0100	*(an ++= xm)	Postaddition of index register, with modify
0101	*(an – –= xm)	Postsubtraction of index register, with modify
0110	*(an ++= imm)	Postaddition of immediate, with modify
0111	*(an – –= imm)	Postsubtraction of immediate, with modify
1000	*(an + xm)	Preaddition of index register
1001	*(an – xm)	Presubtraction of index register
1010	*(an + imm)	Preaddition of immediate
1011	*(an – imm)	Presubtraction of immediate
1100	*(an += xm)	Preaddition of index register, with modify
1101	*(an –= xm)	Presubtraction of index register, with modify
1110	*(an += imm)	Preaddition of immediate, with modify
1111	*(an –= imm)	Presubtraction of immediate, with modify

Legend:
an Address register in l/g address unit
imm Immediate offset
xm Index register in same unit as an register

L, e codes

The L and e bits combine to specify the type of parallel transfer performed, as shown in Table 23. For the local transfer, L and e are bits 21 and 31, respectively. For the global transfer, L and e are bits 17 and 9, respectively.

Table 23. Parallel Transfer Type

L	e	PARALLEL TRANSFER
1	0	Zero-extend load
1	1	Sign-extend load
0	0	Store
0	1	Address unit arithmetic

size codes

The size code specifies the data transfer size. For field moves (parallel transfer format 3), only byte and halfword data sizes are valid, as shown in Table 24.

Table 24. Transfer Data Size

CODING	DATA SIZE
00	Byte (8 bits)
01	Halfword (16 bits)
10	Word (32 bits)
11	Reserved

relative-addressing mode codes

The Lrm and Grm opcode fields allow the local-address or global-address units, respectively, to select PP-relative addressing as shown in Table 25.

Table 25. Relative-Addressing Mode Codes

CODING	RELATIVE-ADDRESSING MODE
00	Normal (absolute addressing)
01	Reserved
10	PP-relative dba
11	PP-relative pba

Legend:
dba – Data RAM 0 base is base address
pba – Paramater RAM base is base address

condition codes

In the four conditional parallel transfer opcodes (formats 7–10), the condition code field specifies one of 16 condition codes to be applied to the data-unit operation source, data-unit result, or global transfer based on the setting of the c, r, and g bits, respectively. Table 26 shows the condition codes. For the 32-bit immediate data unit opcode (format D), the condition applies to the data-unit result only.

Table 26. Condition Codes

CONDITION BITS				MNEMONIC	DESCRIPTION	STATUS BIT COMBINATION
35	34	33	32			
0	0	0	0	u	Unconditional (default)	None
0	0	0	1	p	Positive	~N & ~Z
0	0	1	0	ls	Lower than or same	~C | Z
0	0	1	1	hi	Higher than	C & ~Z
0	1	0	0	lt	Less than	(N & ~V) | (~N & V)
0	1	0	1	le	Less than or equal	(N & ~V) | (~N & V) | Z
0	1	1	0	ge	Greater than or equal	(N & V) | (~N & ~V)
0	1	1	1	gt	Greater than	(N & V & ~Z) | (~N & ~V & ~Z)
1	0	0	0	hs, c	Higher than or same, carry	C
1	0	0	1	lo, nc	Lower than, no carry	~C
1	0	1	0	eq, z	Equal, zero	Z
1	0	1	1	ne, nz	Not equal, not zero	~Z
1	1	0	0	v	Overflow	V
1	1	0	1	nv	No overflow	~V
1	1	1	0	n	Negative	N
1	1	1	1	nn	Nonnegative	~N

EALU operations

Extended ALU (EALU) operations allow the execution of more advanced ALU functions than those specified in the base set ALU opcodes. The opcode for EALU instructions contains the operands for the operation while the d0 register extends the opcode by specifying the EALU operation to be performed. The format of d0 for EALU operations is shown in Figure 24.

EALU Boolean functions

EALU operations support all 256 Boolean ALU functions plus the flexibility to add 1 or a carry-in to Boolean sum. The Boolean function performed by the ALU are shown below and in Table 27.

```
(F0 & (~A & ~B & ~C)) | (F1 & (A & ~B & ~C)) | (F2 & (~A & B & ~C))
| (F3 & (A & B & ~C)) | (F4 & (~A & ~B & C)) | (F5 & (A & ~B & C))
| (F6 & (~A & B & C)) | (F7 & (A & B & C)) [+1 | +cin]
```

Table 27. EALU Boolean Function Codes

d0 BIT	ALU FUNCTION SIGNAL	PRODUCT TERM
26	F7	A & B & C
25	F6	~A & B & C
24	F5	A & ~B & C
23	F4	~A & ~B & C
22	F3	A & B & ~C
21	F2	~A & B & ~C
20	F1	A & ~B & ~C
19	F0	~A & ~B & ~C

EALU arithmetic functions

EALU operations support all 256 arithmetic functions provided by the three-input ALU plus the flexibility to add 1 or a carry-in to the result. The arithmetic function performed by the ALU is:

```
f(A,B,C) = A & f1(B,C) + f2(B,C) [+1 | cin]
```

f1(B,C) and f2(B,C) are independent Boolean combinations of the B and C ALU inputs. The ALU function is specified by selecting the desired f1 and f2 subfunction and then XORing the f1 and f2 code from Table 28 to create the ALU function code for bits 19–26 of d0. Additional operations such as absolute values and signed shifts can be performed using d0 bits which control the ALU function based on the sign of one of the inputs.

Table 28. ALU f1(B,C) and f2(B,C) Subfunctions

f1 CODE	f2 CODE	SUBFUNCTION	COMMON USAGE
00	00	0	Zero the term
AA	FF	–1	–1 (All 1s)
88	CC	B	B
22	33	–B –1	Negate B
A0	F0	C	C
0A	0F	–C –1	Negate C
80	C0	B & C	Force bits in B to 0 where bits in C are 0
2A	3F	–(B & C) – 1	Force bits in B to 0 where bits in C are 0 and negate
A8	FC	B | C	Force bits in B to 1 where bits in C are 1
02	03	–(B | C) – 1	Force bits in B to 1 where bits in C are 1 and negate
08	0C	B & ~C	Force bits in B to 0 where bits in C are 1
A2	F3	–(B & ~C) –1	Force bits in B to 0 where bits in C are 1 and negate
8A	CF	B | ~C	Force bits in B to 1 where bits in C are 0
20	30	–(B | ~C) –1	Force bits in B to 1 where bits in C are 0 and negate
28	3C	(B & ~C) | ((–B – 1) & C)	Choose B if C = all 0s and –B if C = all 1s
82	C3	(B & C) | ((–B – 1) & ~C)	Choose B if C = all 1s and –B if C = all 0s

video controller architecture

The video controller (VC) provides a method for handling the video or graphics capture, or display portions of a TMS320C80 system. It provides simultaneous control over two independent capture or display systems and frame grabber or frame buffer image storage.

VC functional block diagram

Figure 46 shows a functional block diagram of the video controller. Key features of the VC include:

- Dual-frame timers
  - Independent or locked operation
  - Programmable horizontal and vertical timing
  - Separate or composite sync and blanking control
  - Synchronization to external timing signals
  - Interlaced or noninterlaced frame control
  - Virtually limitless screen resolutions
- Programmable timing and control registers
- Programmable line interrupt to MP
- Shift register transfer (SRT) controller
  - Generates VRAM serial register transfer requests to the TC
  - Tracks VRAM tap point and schedules midline reloads
  - Generates packet-transfer requests for DRAM-based buffer updates
  - Supports two display or capture buffers

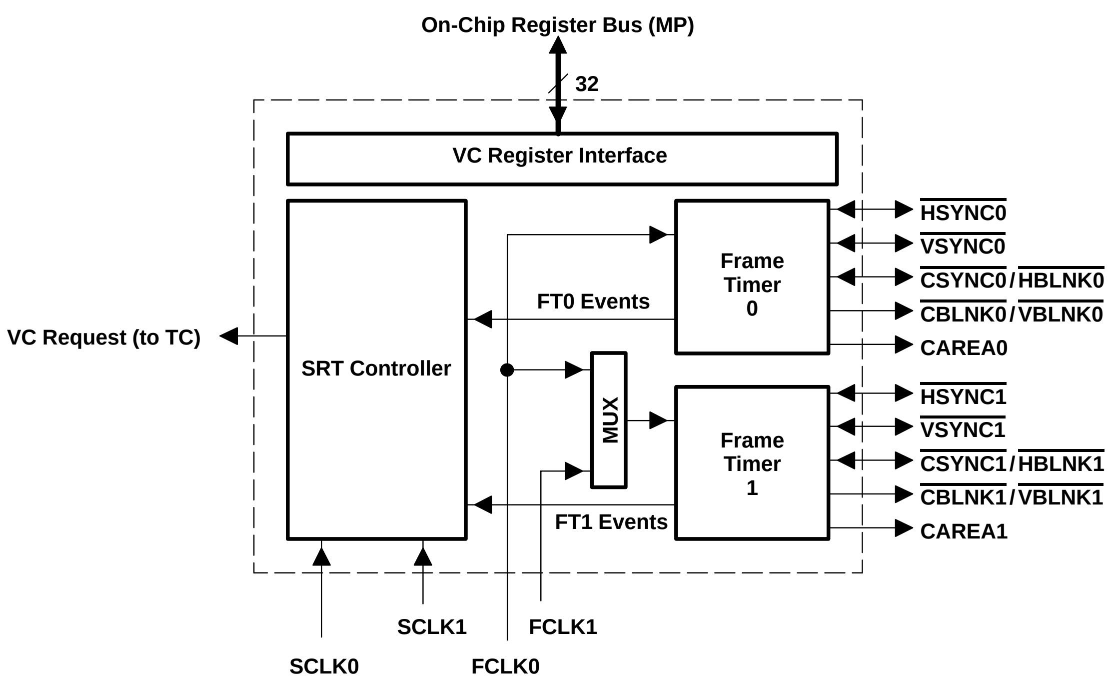


Figure 46. VC Block Diagram

frame-timer registers

Each frame timer has twenty-one 16-bit registers to control its horizontal and vertical timing signals. The registers are on-chip memory-mapped registers accessible by the MP only. Each horizontal/vertical register pair can be accessed as a single 32-bit quantity. The register map for Frame-Timer 0 is shown in Figure 47. The Frame-Timer 1 register map is shown in Figure 48.

Register	Address	Register	Address
		FTCTL0	0x01820200
SETVCT0	0x01820206	SETHCT0	0x01820204
VFTINT0	0x0182020A	HESERR0	0x01820208
VESYNC0	0x0182020E	HESYNC0	0x0182020C
VEBLNK0	0x01820212	HEBLNK0	0x01820210
VSAREA0	0x01820216	HSAREA0	0x01820214
VEAREA0	0x0182021A	HEAREA0	0x01820218
VSBLNK0	0x0182021E	HSBLNK0	0x0182021C
VTOTAL0	0x01820222	HTOTAL0	0x01820220
		HALINE0	0x01820224
		HBLINE0	0x01820228
VCOUNT0	0x0182023E	HCOUNT0	0x0182023C

Figure 47. Frame-Timer 0 Register Map

Register	Address	Register	Address
		FTCTL1	0x01820240
SETVCT1	0x01820246	SETHCT1	0x01820244
VFTINT1	0x0182024A	HESERR1	0x01820248
VESYNC1	0x0182024E	HESYNC1	0x0182024C
VEBLNK1	0x01820252	HEBLNK1	0x01820250
VSAREA1	0x01820256	HSAREA1	0x01820254
VEAREA1	0x0182025A	HEAREA1	0x01820258
VSBLNK1	0x0182025E	HSBLNK1	0x0182025C
VTOTAL1	0x01820262	HTOTAL1	0x01820260
		HALINE1	0x01820264
		HBLINE1	0x01820268
VCOUNT1	0x0182027E	HCOUNT1	0x0182027C

Figure 48. Frame-Timer 1 Register Map

frame-timer register programming

The register format for the frame-timer control registers is shown in Figure 49. All other registers are 16-bit values. For programming details, see the *TMS320C80 Video Controller User's Guide* (literature number SPRU111).

frame-timer control (FTCTLx) register

The FTCTLx register contains mode bits to determine frame-timer behavior.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FTE	IFD	IIM				SSE	FLE			CPM		VPM		HPM	

FTE Frame-timer enable
IFD Interlaced frame disable
IIM Interlace interrupt mode
SSE Set synchronization enable
FLE Frame lock enable
CPM CSYNC / HBLNK pin mode
00 – CSYNC Hi-Z 10 – CSYNC output
01 – CSYNC input 11 – HBLNK output

VPM VSYNC Pin Mode
00 – Hi-Z 10 – Output
01 – Input 11 – Reserved
HPM HSYNC Pin Mode
00 – Hi-Z 10 – Output
01 – Input 11 – Reserved

Figure 49. FTCTLx Register

SRT controller registers

The SRT controller has two sets of 32-bit registers, one for each of the supported frame memory regions. The location of these registers in on-chip memory-mapped register space is shown in Figure 50.

Register	Address	Register	Address
FMEMCTL0	0x01820300	FMEMCTL1	0x01820340
F1STADR0	0x01820304	F1STADR1	0x01820344
F0STADR0	0x01820308	F0STADR1	0x01820348
LINEINC0	0x0182030C	LINEINC1	0x0182034C
SAMMASK0	0x01820310	SAMMASK1	0x01820350
NEXTADR0	0x01820314	NEXTADR1	0x01820354
CRNTADR0	0x0182033C	CRNTADR1	0x0182037C

Figure 50. SRT Controller Register Map

SRT controller register programming

The register format for the frame memory control registers is shown in Figure 51. All other registers are 32-bit values. For programming details, see the *TMS320C80 Video Controller User's Guide* (literature number SPRU111).

FMEMCTLx Register

The frame memory control (FMEMCTLx) register contains mode bits to determine operation of the associated frame memory.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																	HSS	ILR	PTS			TMS			EMS			UED		FTS	

HSS Half-SAM select
ILR Interlaced line repeat
PTS Packet transfer select
TMS Transfer mode select
00 – Display 10 – Capture
01 – Reserved 11 – Merge capture

EMS Event mode select
00 – sof, line, sam 10 – sof, line
01 – sof, eof, sam 11 – none
UED Unblanked event disable
FTS Frame timer sequencer
00 – ft0 / disabled 10 – ft1 / disabled
01 – ft0 / enabled 11 – ft1 / enabled

Figure 51. FMEMCTLx Register

TC architecture

The transfer controller (TC) is a combined memory controller and DMA (direct memory access) machine. It handles the movement of data within the ’C80 system as requested by the master processor, parallel processors, video controller, and external devices. The transfer controller performs the following data movement and memory control functions:

- MP and PP instruction cache fills
- MP data cache fills and dirty block write-back
- MP and PP direct external accesses (DEAs)
- MP and PP packet transfers
- Externally initiated packet transfers (XPTs)
- VC packet transfers (VCPTs)
- VC shift register transfers (SRTs)
- DRAM / SDRAM refresh
- Host bus request

TC functional block diagram

Figure 52 shows a functional block diagram of the transfer controller. Key features of the TC include:

- Crossbar interface
  - 64-bit data path
  - Single-cycle access
- External memory interface
  - 4G-Byte address range
  - Dynamically configurable memory cycles

    8-, 16-, 32-, or 64-bit bus size
    Selectable memory page size
    Selectable address multiplexing
    Selectable cycle timing
  - Big- or little-endian operation
- Cache, VRAM, refresh controller
  - Programmable refresh rate
  - VRAM block write support
- Independent Src and Dst addressing
  - Autonomous addressing based on packet-transfer parameters
  - Data read and write at different rates
  - Numerous data merging and alignment functions performed during transfer
- Intelligent request prioritization

TC functional block diagram (continued)

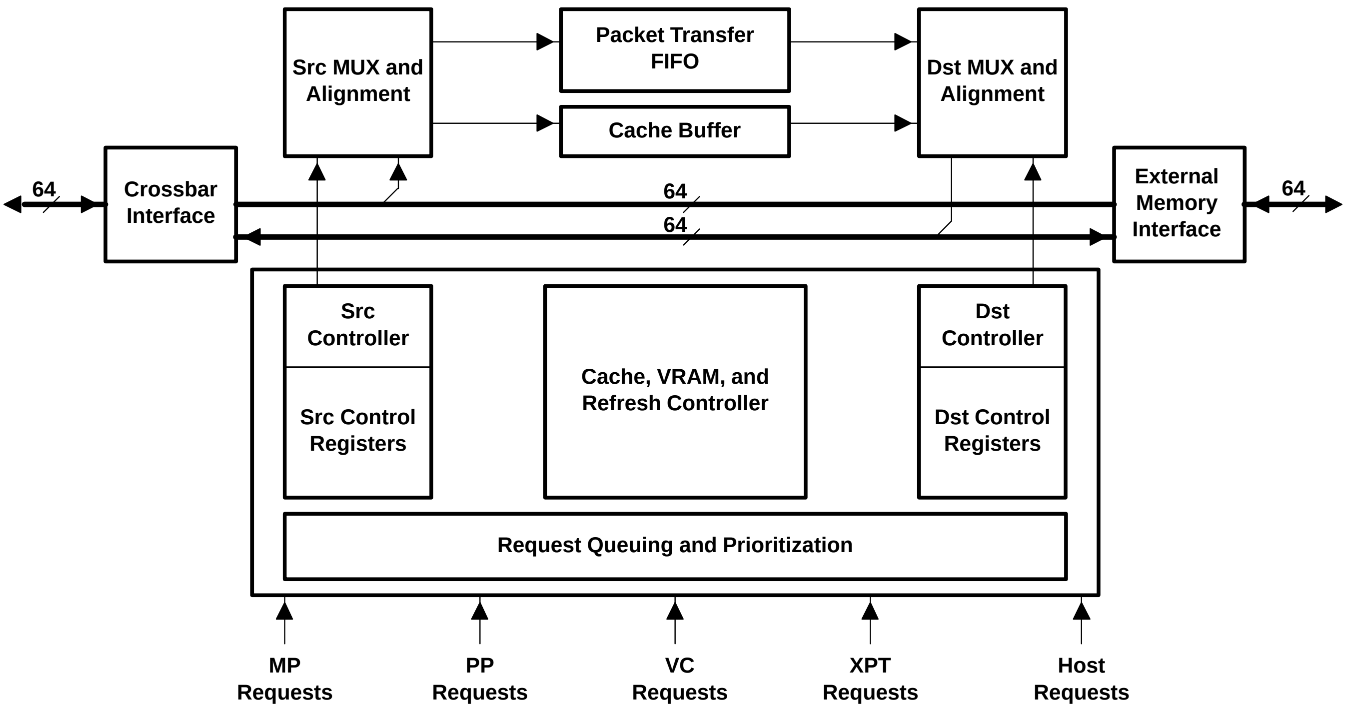


Figure 52. TC Block Diagram

TC registers

The TC contains four on-chip memory-mapped registers accessible by the MP. TC registers are shown in Figure 53.

refresh control (REFCNTL) register (0x01820000)

The REFCNTL register controls refresh cycles.

31–16	15–0
RPARLD	REFRATE

RPARLD **Refresh Pseudo-Address Reload Value** **REFRATE** **Refresh Interval (in clock cycles)**

Figure 53. REFCNTL Register

packet-transfer minimum (PTMIN) register (0x01820004)

The PTMIN register determines the minimum number of cycles that a packet transfer executes before being suspended by a higher priority packet transfer. Figure 54 shows the PTMIN register.

31–24	23–0
	PTMIN

Figure 54. PTMIN Register

PT maximum (PTMAX) register (0x01820008)

The PTMAX register determines the maximum number of cycles after PTMIN has elapsed that a packet transfer executes before timing out. Figure 55 showns the format of the PTMAX register.

31–24	23–0
	PTMAX

Figure 55. PTMAX Register

fault status (FLTSTS) register (0x0182000C)

The FLTSTS register indicates the cause of a memory access fault. Fault status bits are cleared by writing a 1 to the appropriate bit. Figure 56 shows the format of the fault status (FLTSTS) register.

31–28	27	26	25	24	23–20	19	18	17	16	15–8	7–5	4–1	0
	PC	PC	PC	PC		PP	PP	PP	PP		XPT		M
PP #	3	2	1	0	PP#	3	2	1	0				

PC PPx Cache / DEA Fault
PP PPx Packet-Transfer Fault
XPT Faulting XPT
M MP Packet-Transfer Fault

Figure 56. FLTSTS Register

packet-transfer parameters

The most efficient method for data movement in a TMS320C80 system is through the use of packet transfers (PTs). Packet transfers allow the TC to move blocks of data autonomously between a specified src and dst memory region. Requests for the TC to execute a packet transfer may be made by the MP, PPs, VC, or external devices. A packet-transfer parameter table describing the data packet and how it is to be transferred must be programmed in on-chip memory before the transfer is requested. The parameter table formats for long-form and short-form packet transfers are shown in Figure 57.

packet-transfer parameters (continued)

Long-Form Parameter Table

Byte Address	31	0	Word Number
PT	Next Entry Address		0
PT+4	PT Options		1
PT+8	Src Start Address		0
PT+12	Dst Start Address		1
PT+16	Src B Count	Src A Count	0
PT+20	Dst B Count	Dst A Count	1
PT+24	Src C Count		0
PT+28	Dst C Count		1
PT+32	Src B Pitch		0
PT+36	Dst B Pitch		1
PT+40	Src C Pitch		0
PT+44	Dst C Pitch		1
†PT+48	Transparency/Color Word 0		0†
†PT+52	Transparency/Color Word 1		1†
PT+56	Don’t Care		0
PT+60	Don’t Care		1

PT - 64-byte aligned on-chip starting address of parameter table.

† These words are swapped in big-endian mode.

Short-Form Parameter Table

Byte Address	31	0	Word Number
PT	Next Entry Address		0
PT+4	PT Options	Count	1
PT+8	Src Start Address		0
PT+12	Dst Start Address		1

PT - 16-byte aligned on-chip starting address of parameter table.

Figure 57. Packet-Transfer Parameter Table

PT-options field

The PT-options field of the parameter table controls the type of src and dst transfer that the TC performs. The formats of the options field for long-form and short-form packet transfers are shown in Figure 58.

Long-Form PT Options

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	PTS		I	(reserved)		RDC	RDB	RA	RSC	RSB	SF	X	PAM			(reserved)	STM			(reserved)		SUM		(reserved)	DTM			(reserved)		DUM	

S	Stop Bit		
PTS	PT Status		
	00 – Active	10 – Fault on Src	
	01 – Suspended	11 – Fault on Dst	
I	Stop Bit		
RDC	Reverse Dst C Addressing		
RDB	Reverse Dst B Addressing		
RA	Reverse A Addressing		
RSC	Reverse Src C Addressing		
RSB	Reverse Src B Addressing		
SF	Short Form		
X	Exchange Src and Dst Parameters		

PAM	PT Access Mode		
	000 – Normal	100 – 8-Bit Transparency	
	001 – PDT	101 – 16-Bit Transparency	
	010 – Block Write	110 – 32-Bit Transparency	
	011 – SRT	111 – 64-Bit Transparency	
STM/DTM	Src/Dst Transfer Mode		
	000 – Dimensioned	100 – Variable, Delta-Guided	
	001 – Fill (Src Only)	101 – Variable, Offset-Guided	
	010 – Reserved	110 – Fixed, Delta-Guided	
	011 – LUT (Src Only)	111 – Fixed, Offset-Guided	
SUM/DTM	Src/Dst Update Mode		
	00 – None	01 – Add C Pitch	
	01 – Add B Pitch	11 – Add C Pitch/Reverse	

Short-Form PT Options

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	PTS		I	(reserved)				RA	(reserved)		SF	X	PAM			Count															

S	Stop Bit		
PTS	PT Status		
	00 – Active	10 – Fault on Src	
	01 – Suspended	11 – Fault on Dst	
I	Stop Bit		
RA	Reverse Addressing		

RA	Reverse Addressing		
X	Exchange Src & Dst Parameters		
PAM	PT Access Mode		
	000 – Normal	100 – Reserved	
	001 – PDT	101 – Reserved	
	010 – Reserved	110 – Reserved	
	011 – SRT	111 – Reserved	

Figure 58. PT-Options Field

LOCAL MEMORY INTERFACE

status codes

Status codes are output on STATUS[5:0] to describe the cycle being performed. During row time, the STATUS[5:0] pins indicate the type of cycle being performed. The cycle type can be latched using $\overline{RL}$ or $\overline{RAS}$ and used by external logic to perform memory bank decoding or to enable special hardware features. During column time, the STATUS[5:0] pins indicate the requesting processor or special column information. See Table 29 for a listing of the Row-time status codes and Table 30 for a listing of column-time status codes.

Table 29. Row-Time Status Codes

STATUS[5:0]	CYCLE TYPE	STATUS[5:0]	CYCLE TYPE
0 0 0 0 0 0	Normal Read	1 0 0 0 0 0	Reserved
0 0 0 0 0 1	Normal Write	1 0 0 0 0 1	Reserved
0 0 0 0 1 0	Refresh	1 0 0 0 1 0	Reserved
0 0 0 0 1 1	SDRAM DCAB	1 0 0 0 1 1	Reserved
0 0 0 1 0 0	Peripheral Device PT Read	1 0 0 1 0 0	XPT1 Read
0 0 0 1 0 1	Peripheral Device PT Write	1 0 0 1 0 1	XPT1 Write
0 0 0 1 1 0	Reserved	1 0 0 1 1 0	XPT1 PDPT Read
0 0 0 1 1 1	Reserved	1 0 0 1 1 1	XPT1 PDPT Write
0 0 1 0 0 0	Reserved	1 0 1 0 0 0	XPT2 Read
0 0 1 0 0 1	Block Write PT	1 0 1 0 0 1	XPT2 Write
0 0 1 0 1 0	Reserved	1 0 1 0 1 0	XPT2 PDPT Read
0 0 1 0 1 1	Reserved	1 0 1 0 1 1	XPT2 PDPT Write
0 0 1 1 0 0	SDRAM MRS	1 0 1 1 0 0	XPT3 Read
0 0 1 1 0 1	Load Color Register	1 0 1 1 0 1	XPT3 Write
0 0 1 1 1 0	Reserved	1 0 1 1 1 0	XPT3 PDPT Read
0 0 1 1 1 1	Reserved	1 0 1 1 1 1	XPT3 PDPT Write
0 1 0 0 0 0	Frame 0 Read Transfer	1 1 0 0 0 0	XPT4/SAM1 Read
0 1 0 0 0 1	Frame 0 Write Transfer	1 1 0 0 0 1	XPT4/SAM1 Write
0 1 0 0 1 0	Frame 0 Split Read Transfer	1 1 0 0 1 0	XPT4/SAM1 PDPT Read
0 1 0 0 1 1	Frame 0 Split Write Transfer	1 1 0 0 1 1	XPT4/SAM1 PDPT Write
0 1 0 1 0 0	Frame 1 Read Transfer	1 1 0 1 0 0	XPT5/SOF1 Read
0 1 0 1 0 1	Frame 1 Write Transfer	1 1 0 1 0 1	XPT5/SOF1 Write
0 1 0 1 1 0	Frame 1 Split Read Transfer	1 1 0 1 1 0	XPT5/SOF1 PDPT Read
0 1 0 1 1 1	Frame 1 Split Write Transfer	1 1 0 1 1 1	XPT5/SOF1 PDPT Write
0 1 1 0 0 0	Reserved	1 1 1 0 0 0	XPT6/SAM0 Read
0 1 1 0 0 1	Reserved	1 1 1 0 0 1	XPT6/SAM0 Write
0 1 1 0 1 0	Reserved	1 1 1 0 1 0	XPT6/SAM0 PDPT Read
0 1 1 0 1 1	Reserved	1 1 1 0 1 1	XPT6/SAM0 PDPT Write
0 1 1 1 0 0	PT Read Transfer	1 1 1 1 0 0	XPT7/SOF0 Read
0 1 1 1 0 1	PT Write Transfer	1 1 1 1 0 1	XPT7/SOF0 Write
0 1 1 1 1 0	Reserved	1 1 1 1 1 0	XPT7/SOF0 PDPT Read
0 1 1 1 1 1	Idle	1 1 1 1 1 1	XPT7/SOF0 PDPT Write

status codes (continued)

Table 30. Column-Time Status Codes

STATUS[5:0]	CYCLE TYPE	STATUS[5:0]	CYCLE TYPE
0 0 0 0 0 0	PP0 Low Priority Packet Transfer	1 0 0 0 0 0	Reserved
0 0 0 0 0 1	PP0 High Priority Packet Transfer	1 0 0 0 0 1	Reserved
0 0 0 0 1 0	PP0 Instruction Cache	1 0 0 0 1 0	Reserved
0 0 0 0 1 1	PP0 DEA	1 0 0 0 1 1	Reserved
0 0 0 1 0 0	PP1 Low Priority Packet Transfer	1 0 0 1 0 0	Reserved
0 0 0 1 0 1	PP1 High Priority Packet Transfer	1 0 0 1 0 1	Reserved
0 0 0 1 1 0	PP1 Instruction Cache	1 0 0 1 1 0	Reserved
0 0 0 1 1 1	PP1 DEA	1 0 0 1 1 1	Reserved
0 0 1 0 0 0	PP2 Low Priority Packet Transfer	1 0 1 0 0 0	Reserved
0 0 1 0 0 1	PP2 High Priority Packet Transfer	1 0 1 0 0 1	Reserved
0 0 1 0 1 0	PP2 Instruction Cache	1 0 1 0 1 0	Reserved
0 0 1 0 1 1	PP2 DEA	1 0 1 0 1 1	Reserved
0 0 1 1 0 0	PP3 Low Priority Packet Transfer	1 0 1 1 0 0	Reserved
0 0 1 1 0 1	PP3 High Priority Packet Transfer	1 0 1 1 0 1	Reserved
0 0 1 1 1 0	PP3 Instruction Cache	1 0 1 1 1 0	Reserved
0 0 1 1 1 1	PP3 DEA	1 0 1 1 1 1	Reserved
0 1 0 0 0 0	MP Low Priority Packet Transfer	1 1 0 0 0 0	Reserved
0 1 0 0 0 1	MP High Priority Packet Transfer	1 1 0 0 0 1	Reserved
0 1 0 0 1 0	MP Urgent Packet Transfer (Low)	1 1 0 0 1 0	Reserved
0 1 0 0 1 1	MP Urgent Packet Transfer (High)	1 1 0 0 1 1	Reserved
0 1 0 1 0 0	XPT / VCPT in Progress	1 1 0 1 0 0	Reserved
0 1 0 1 0 1	XPT / VCPT Complete	1 1 0 1 0 1	Reserved
0 1 0 1 1 0	MP Instruction Cache (Low)	1 1 0 1 1 0	Reserved
0 1 0 1 1 1	MP Instruction Cache (High)	1 1 0 1 1 1	Reserved
0 1 1 0 0 0	MP DEA (Low)	1 1 1 0 0 0	Reserved
0 1 1 0 0 1	MP DEA (High)	1 1 1 0 0 1	Reserved
0 1 1 0 1 0	MP Data Cache (Low)	1 1 1 0 1 0	Reserved
0 1 1 0 1 1	MP Data Cache (High)	1 1 1 0 1 1	Reserved
0 1 1 1 0 0	Frame 0	1 1 1 1 0 0	Reserved
0 1 1 1 0 1	Frame 1	1 1 1 1 0 1	Reserved
0 1 1 1 1 0	Refresh	1 1 1 1 1 0	Reserved
0 1 1 1 1 1	Idle	1 1 1 1 1 1	Write Drain / SDRAM DCAB

Low – MP operating in low (normal) priority mode
High – MP operating in high priority mode

address multiplexing

To support various RAM devices, the TMS320C80 can provide multiplexed row and column addresses on its address bus. A full 32-bit address is always output at row time. The alignment of column addresses is configured by the value input on the AS[2:0] pins at row time (see Figure 59).

A[31:0] Pins

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Row Time

A[31:0] Pins

AS [2:0]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	x	x	x	x	x	2	1	0
010	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	x	x	x	x	x	x	2	1	0
011	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	x	x	x	x	x	x	x	2	1	0
100	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	x	x	x	x	x	x	x	x	2	1	0
101	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	x	x	x	x	x	x	x	x	x	2	1	0
110	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	x	x	x	x	x	x	x	x	x	x	2	1	0
111	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	x	x	x	x	x	x	x	x	x	x	x	2	1	0

Column Time

Figure 59. Address Multiplexing

dynamic bus sizing

The ’C80 supports data bus sizes of 8, 16, 32, or 64 bits as shown in Table 31. The value input on the BS[1:0] pins at row time indicates the bus size of the addressed memory. This determines the maximum number of bytes which the ’C80 can transfer during each column access. If the number of bytes to be transferred exceeds the bus size, multiple accesses are performed automatically to complete the transfer.

Table 31. Bus Size Selection

BS[1:0]	BUS SIZE
0 0	8 bits
0 1	16 bits
1 0	32 bits
1 1	64 bits

The selected bus size also determines which portion of the data bus is used for the transfer. For 64-bit memory, the entire data bus is used. For 32-bit memory, D[31:0] are used in little-endian mode and D[63:32] are used in big-endian mode. 16-bit buses use D[15:0] and D[63:48] and 8-bit buses use D[7:0] and D[63:56] for little- and big-endian modes, respectively. The ’C80 always aligns data to the proper portion of the bus and activates the appropriate $\overline{\text{CAS}}$/DQM strobes to ensure that only valid bytes are transferred.

cycle time selection

The 'C80 supports eight basic sets of memory timings to support various memory types directly. The cycle timing is selected by the value input on the CT[2:0] and $\overline{\text{UTIME}}$ pins at row time. The selected timing remains in effect until the next row access. See Table 32 for Cycle-timing selections.

Table 32. Cycle-Timing Selection

$\overline{\text{UTIME}}$	CT[2:0]			MEMORY TIMING
0	0	0	0	Reserved
0	0	0	1	SDRAM: burst length 1, read latency 4
0	0	1	0	Reserved
0	0	1	1	SDRAM: burst length 2, read latency 4
0	1	0	0	User timed DRAM: pipelined 1 cycle / column
0	1	0	1	User timed DRAM: 1 cycle / column
0	1	1	0	User timed DRAM: 2 cycle / column
0	1	1	1	User timed DRAM: 3 cycle / column
1	0	0	0	SDRAM: burst length 1, read latency 2
1	0	0	1	SDRAM: burst length 1, read latency 3
1	0	1	0	SDRAM: burst length 2, read latency 2
1	0	1	1	SDRAM: burst length 2, read latency 3
1	1	0	0	DRAM: pipelined 1 cycle / column
1	1	0	1	DRAM: 1 cycle / column
1	1	1	0	DRAM: 2 cycle / column
1	1	1	1	DRAM: 3 cycle / column

page sizing

Whenever an external memory access occurs, the TC records the 26 most significant bits of the address in its internal LASTPAGE register. The address of each subsequent (column) access is compared to this value. The page size value input on the PS[3:0] pins determines which bits of LASTPAGE are used for this comparison. If a difference exists between the enabled LASTPAGE bits and the corresponding bits of the next access then the page has changed and the next memory access begins with a new row-address cycle (see Table 33).

page sizing (continued)

Table 33. Page-Size Selection

PS[3:0]	ADDRESS BITS COMPARED	PAGE SIZE (BYTES)
0 0 0 0	A(31:6)	64
0 0 0 1	A(31:7)	128
0 0 1 0	A(31:8)	256
0 0 1 1	A(31:9)	512
0 1 0 0	A(31:10)	1K
0 1 0 1	A(31:18)	256K
0 1 1 0	A(31:19)	512K
0 1 1 1	A(31:20)	1M
1 0 0 0	A(31:0)	1–8†
1 0 0 1	A(31:11)	2K
1 0 1 0	A(31:12)	4K
1 0 1 1	A(31:13)	8K
1 1 0 0	A(31:14)	16K
1 1 0 1	A(31:15)	32K
1 1 1 0	A(31:16)	64K
1 1 1 1	A(31:17)	128K

† PS[3:0] = 1000 disables page-mode cycles so that the effective page size is the same as the bus size

block write support

The TMS320C80 supports three modes of VRAM block write. The block-write mode is dynamically selectable so that software may specify block writes without knowing what type of block write the addressed memory supports. Block writes are supported only for 64-bit buses. During block-write and load-color-register cycles, the BS[1:0] inputs determine which block mode will be used (see Table 34).

Table 34. Block-Write Selection

BS[1:0]	BLOCK-WRITE MODE
0 0	Simulated
0 1	Reserved
1 0	4x
1 1	8x

SDRAM support

The TMS320C80 provides direct support for synchronous DRAM (SDRAM), VRAM (SVRAM), and graphics RAM (SGRAM). During ’C80 power-up refresh cycles, the external system must signal the presence of these memories by inputting a CT2 value of 0. This causes the ’C80 to perform special deactivate (DCAB) and mode register set (MRS) commands to initialize the synchronous RAMs. Figure 60 shows the MRS value generated by the ’C80. Note that read latency 4 timing programs the mode register for a read latency of 3. See Figure 60 for a listing of MRS values.

SDRAM support (continued)

SDRAM Mode Register Bit	11	10	9	8	7	6	5	4	3	2	1	0
Meaning			WB	0	0	Read Latency			S/I	Burst Length		
Value	0	0	0	0	0	!($\overline{\text{UTIME}}$)	$\overline{\text{UTIME}}$	$\overline{\text{UTIME}}$&CT0	0	0	0	CT1

$\overline{\text{UTIME}}$, CT0, CT1 values as input at the start of the MRS cycle

Figure 60. MRS Value

Because the MRS register is programmed through the SDRAM address inputs, the alignment of the MRS data to the ’C80 logical-address bits is adjusted for the bus size (see Figure 61). The appearance of the MRS bits on the ’C80 physical-address bus is dependent on the address multiplexing as selected by the AS[2:0] inputs.

	’C80 LOGICAL ADDRESS BITS															
BS[1:0]	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0 0	X	X	X	X	11	10	9	8	7	6	5	4	3	2	1	0
0 1	X	X	X	11	10	9	8	7	6	5	4	3	2	1	0	X
1 0	X	X	11	10	9	8	7	6	5	4	3	2	1	0	X	X
1 1	X	11	10	9	8	7	6	5	4	3	2	1	0	X	X	X

Figure 61. MRS Value Alignment

memory cycles

TMS320C80 external memory cycles are generated by the TC’s external memory controller. The controller’s state machine generates a sequence of states which define the transition of the memory interface signals. The state sequence is dependent on the cycle timing selected for the memory access being performed as shown in Figure 62. Memory cycles consist of row states and the column pipeline (see Figure 62).

memory cycles (continued)

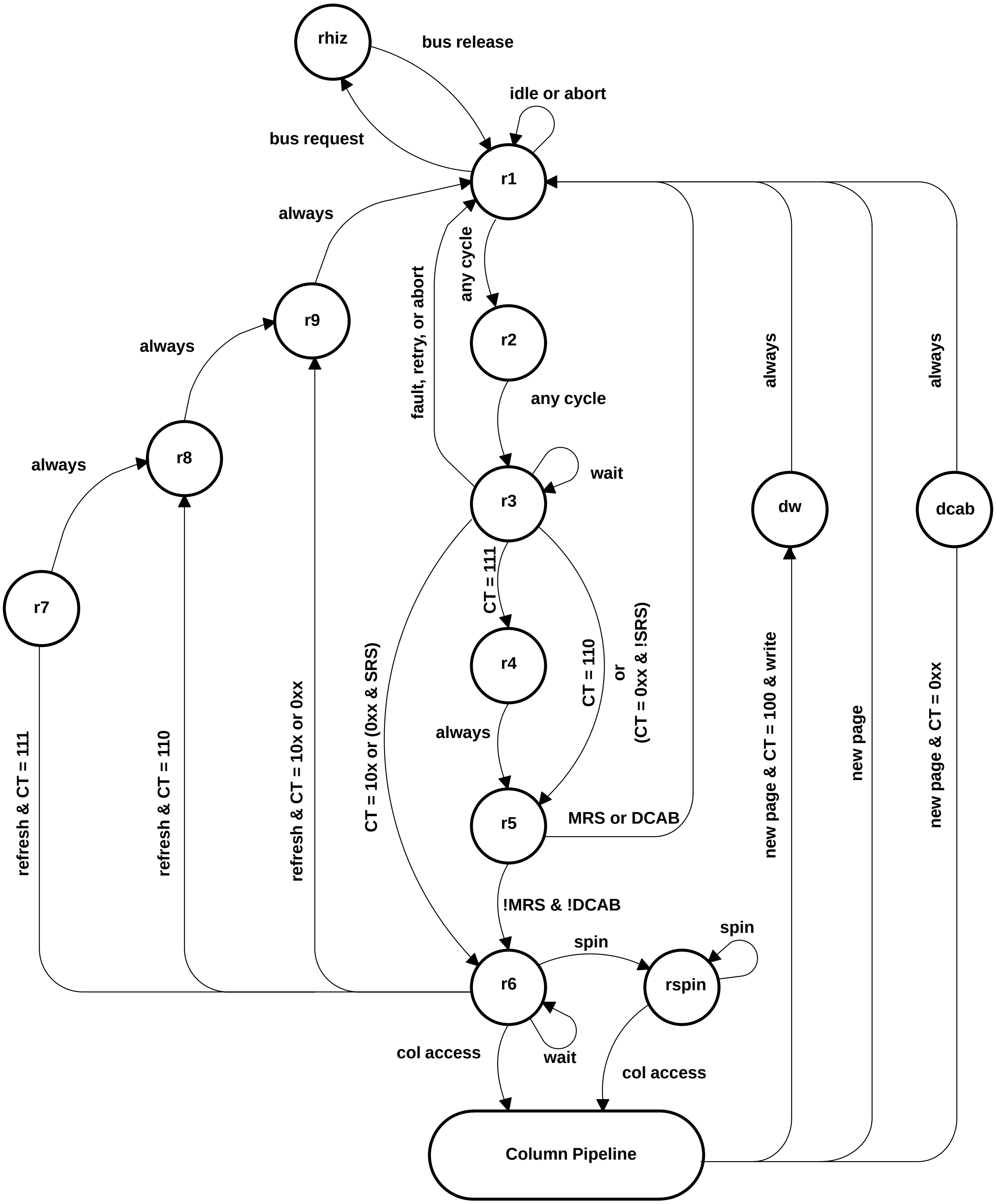


Figure 62. Memory Cycle State Diagram

row states

The row states make up the row time of each memory access. They occur when each new page access begins. The transition indicators determine the conditions that cause transitions to another state. See Table 35 and Table 36.

Table 35. Row States

STATE	DESCRIPTION
r1	Beginning state for all memory accesses. Outputs row address (A[31:0]) and cycle type (STATUS[5:0]) and drives control signals to their inactive state
r2	Common to all memory accesses. Asserts RL and drives DDIN according to the data transfer direction. AS[2:0], BS[1:0], CT[2:0], PS[3:0] and UTIME inputs are sampled
r3	Common to all memory accesses. DBEN is driven to its active level. For non-SDRAM, W, TRG / CAS, and DSF are driven to their active levels and for non-SDRAM refreshes, all CAS / DQM strobes are activated. FAULT, READY, and RETRY inputs are sampled.
r4	Inserted for 3 cycle / column accesses (CT=111) only. No signal transitions occur. RETRY input is sampled.
r5	Common to SDRAM and 2 or 3 cycle / column accesses (CT=0xx or 11x). RAS is driven low. W is driven low for DCAB and MRS cycles and TRG / CAS is driven low for MRS and SDRAM refresh cycles.
r6	Common to all memory accesses. For SDRAM cycles, RAS, TRG / CAS, and W are driven high. For non-SDRAM, RAS is driven low (if not already) and W, TRG / CAS, and DSF are driven to their appropriate levels. DBEN is driven low and READY and RETRY are sampled.
rspin	Additional state to allow TC column time pipeline to load. No signal transitions occur. RETRY is sampled. The rspin state can, on occasion, repeat multiple times.
r7	Common to 2 and 3 cycle / column refreshes (CT=11x). Processor activity code is output on STATUS[5:0]. RETRY input is sampled.
r8	For 3 cycle / column refreshes only (CT=111). No signal transitions occur. RETRY input is sampled.
r9	Common to all refresh cycles. Processor activity code is output on STATUS[5:0] and RETRY input is sampled.
dw	Occurs for pipelined 1 cycle / column writes only. All CAS / DQM strobes are activated.
dcab	Occurs for SDRAM cycles (CT = 0xx). RAS and W are activated to perform a DCAB command.
rhiz	High impedance state. Occurs during host requests and repeats until bus is released by the host

Table 36. State Transition Indicators

INDICATOR	DESCRIPTION
any cycle	Continuation of current cycle
CT=xxx	State change occurs for indicated CT[2:0] value (as latched in r2 state)
abort	Current cycle aborted by TC in favor of higher priority cycle
fault	FAULT input sampled low (in r3 state), memory access faulted
retry	RETRY input sampled low (in r3 state), row-time retry
wait	READY input sampled low (in r3, r6, or last column state) repeat current state
spin	Internally generated wait state to allow TC pipeline to load
new page	The next access requires a page change (new row access).

external memory timing examples

The following sections contain descriptions of the ’C80 memory cycles and illustrate the signal transitions for those cycles. Memory cycles may be separated into two basic categories; DRAM-type cycles for use with DRAM-like devices, SRAM, and peripherals, and SDRAM-type cycles for use with SDRAM-like devices.

DRAM-type cycles

The DRAM-type cycles are page-mode accesses consisting of a row access followed by one or more column accesses. Column accesses may be one, two, or three clock cycles in length with two and three cycle accesses allowing the insertion of wait states to accommodate slow devices. Idle cycles can occur after necessary column accesses have completed or between column accesses due to “bubbles” in the TC data-flow pipeline. The pipeline diagrams in Figure 63 show the pipeline stages for each access type and when the $\overline{\text{CAS}}$/DQM signal corresponding to the column access is activated.

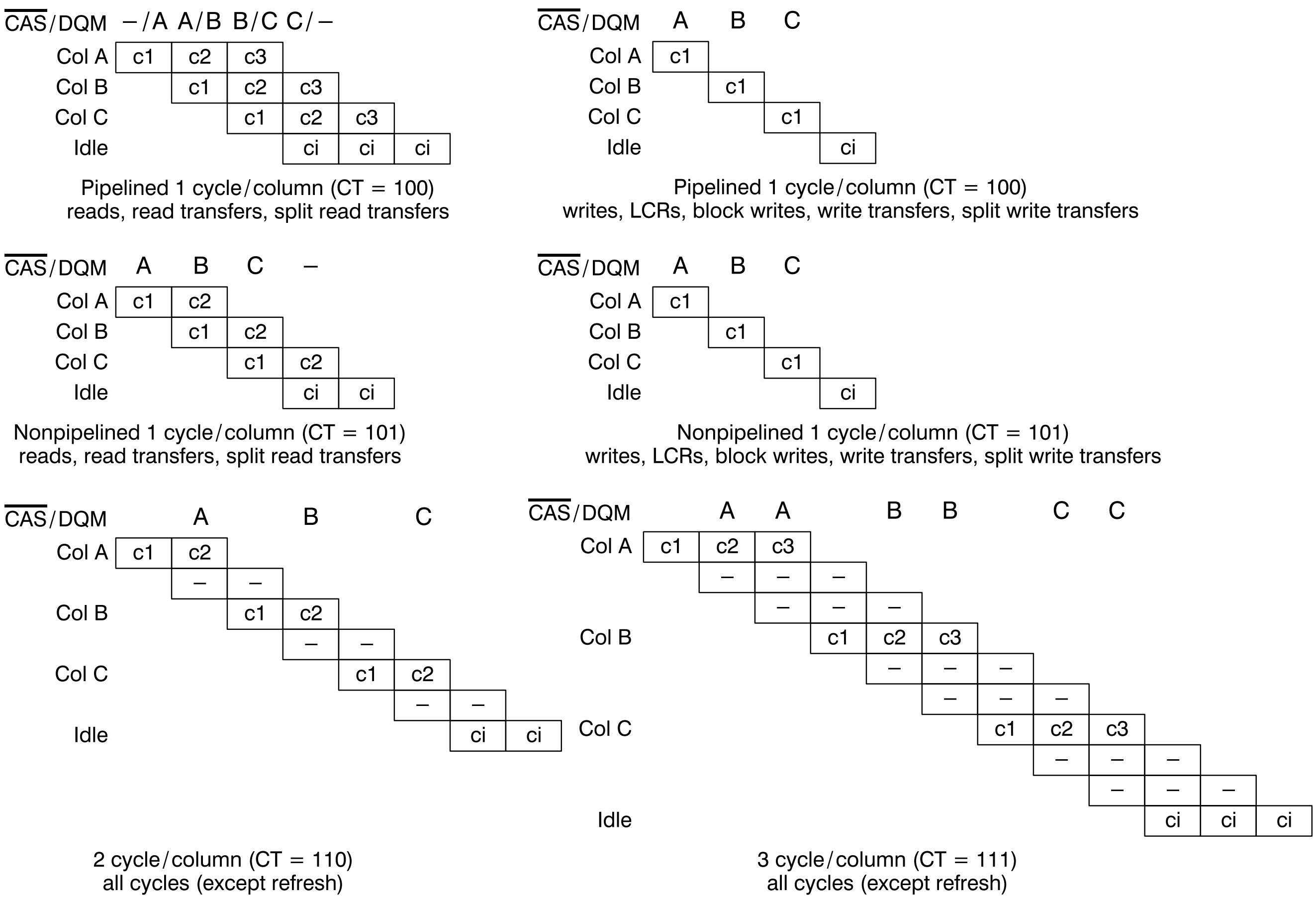


Figure 63. DRAM Cycle Column Pipelines

read cycles

Read cycles transfer data or instructions from external memory to the ’C80. The cycles can occur as a result of a packet transfer, cache request, or DEA request. During the cycle, $\overline{\text{W}}$ is held high, $\overline{\text{TRG}}$/$\overline{\text{CAS}}$ is driven low after $\overline{\text{RAS}}$ to enable memory output drivers and $\overline{\text{DDIN}}$ is low so that data transceivers drive into the ’C80. The TC places D[63:0] in high impedance allowing it to be driven by the memory and latches input data during the appropriate column state. The TC always reads 64 bits and extracts and aligns the appropriate bytes. Invalid bytes for bus sizes of less than 64 bits are discarded. During peripheral device packet transfers, $\overline{\text{DBEN}}$ remains high. Read cycles are shown in Figure 64 through Figure 67.

read cycles (continued)

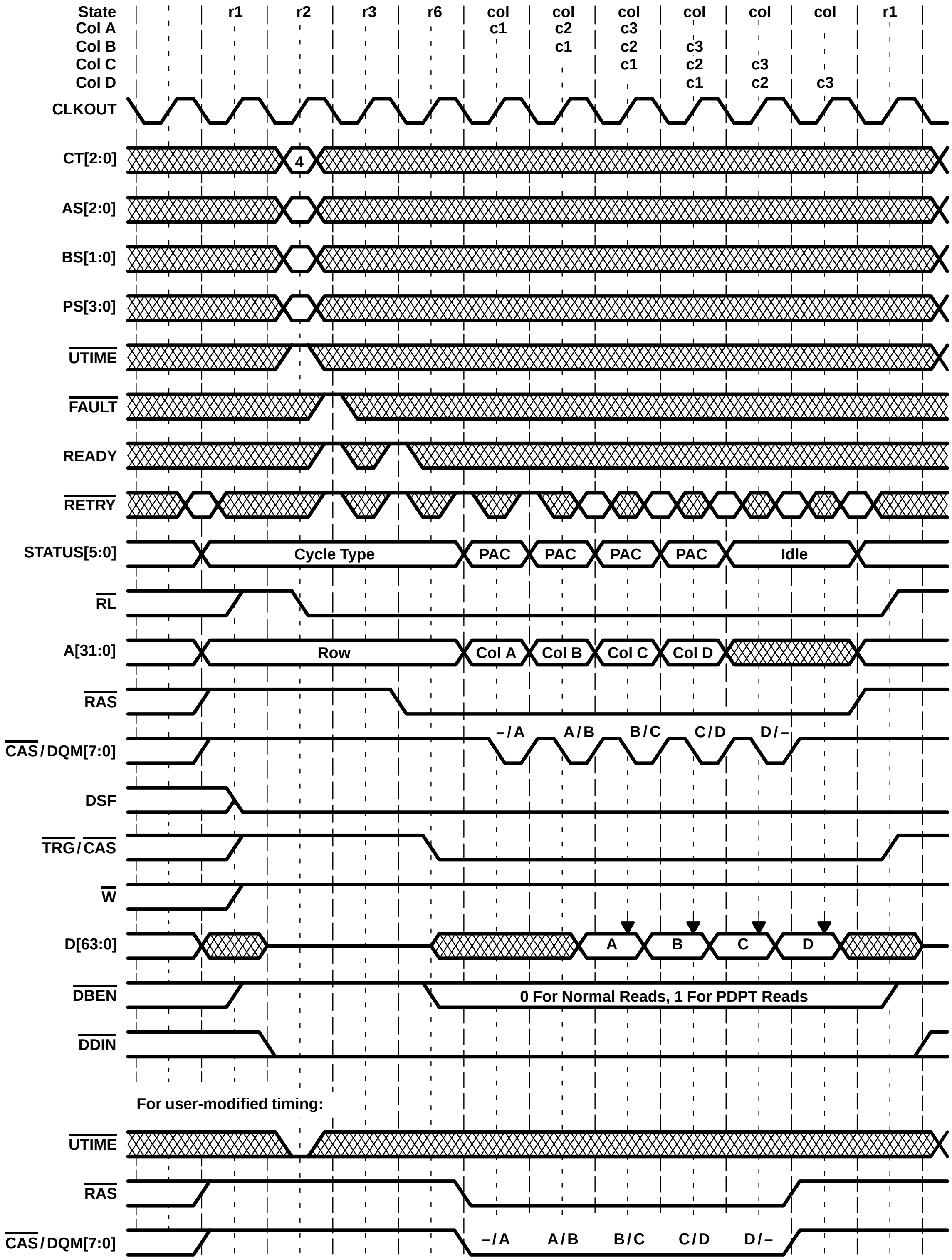


Figure 64. Pipelined 1 Cycle / Column Read-Cycle Timing

read cycles (continued)

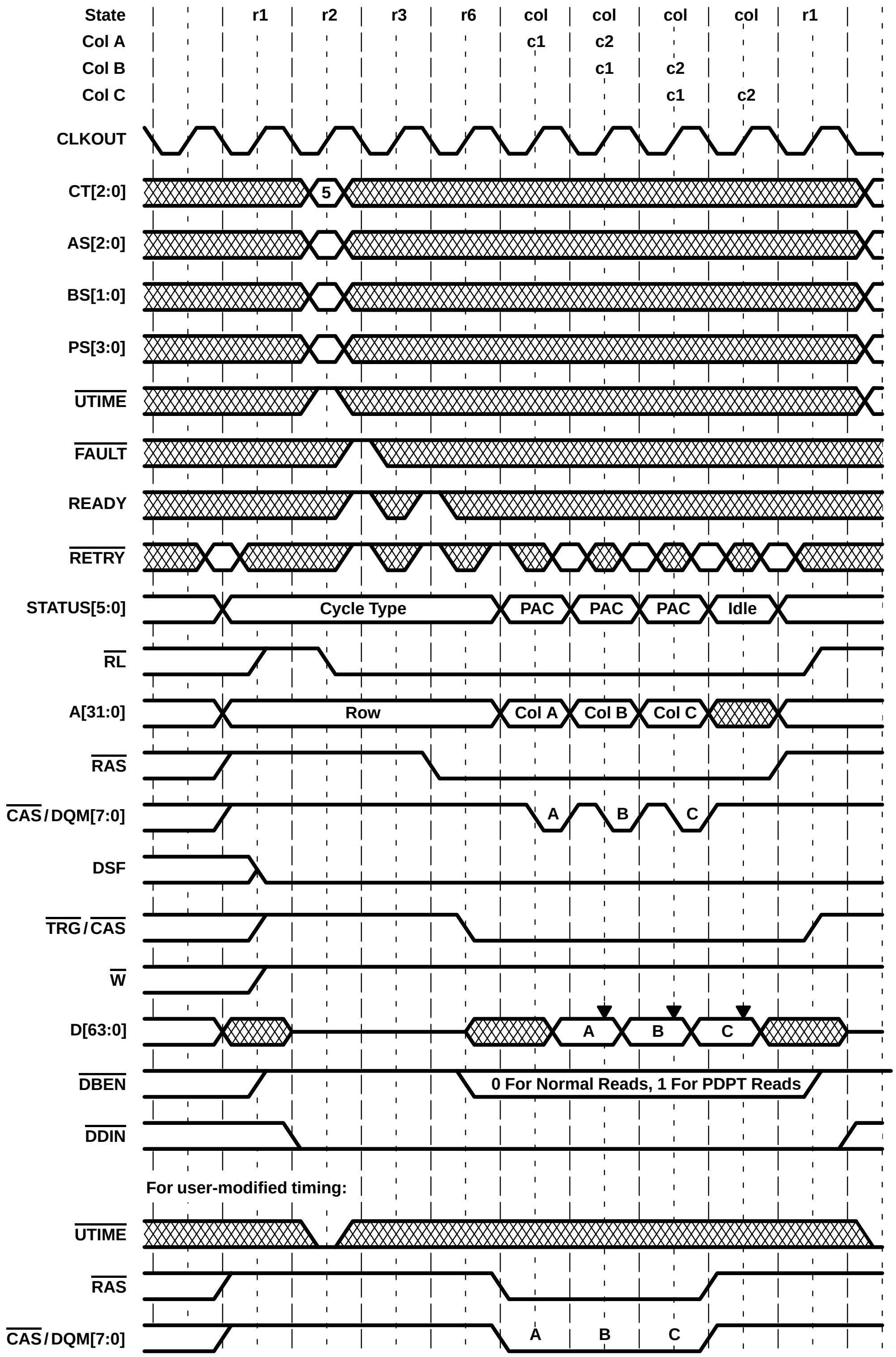
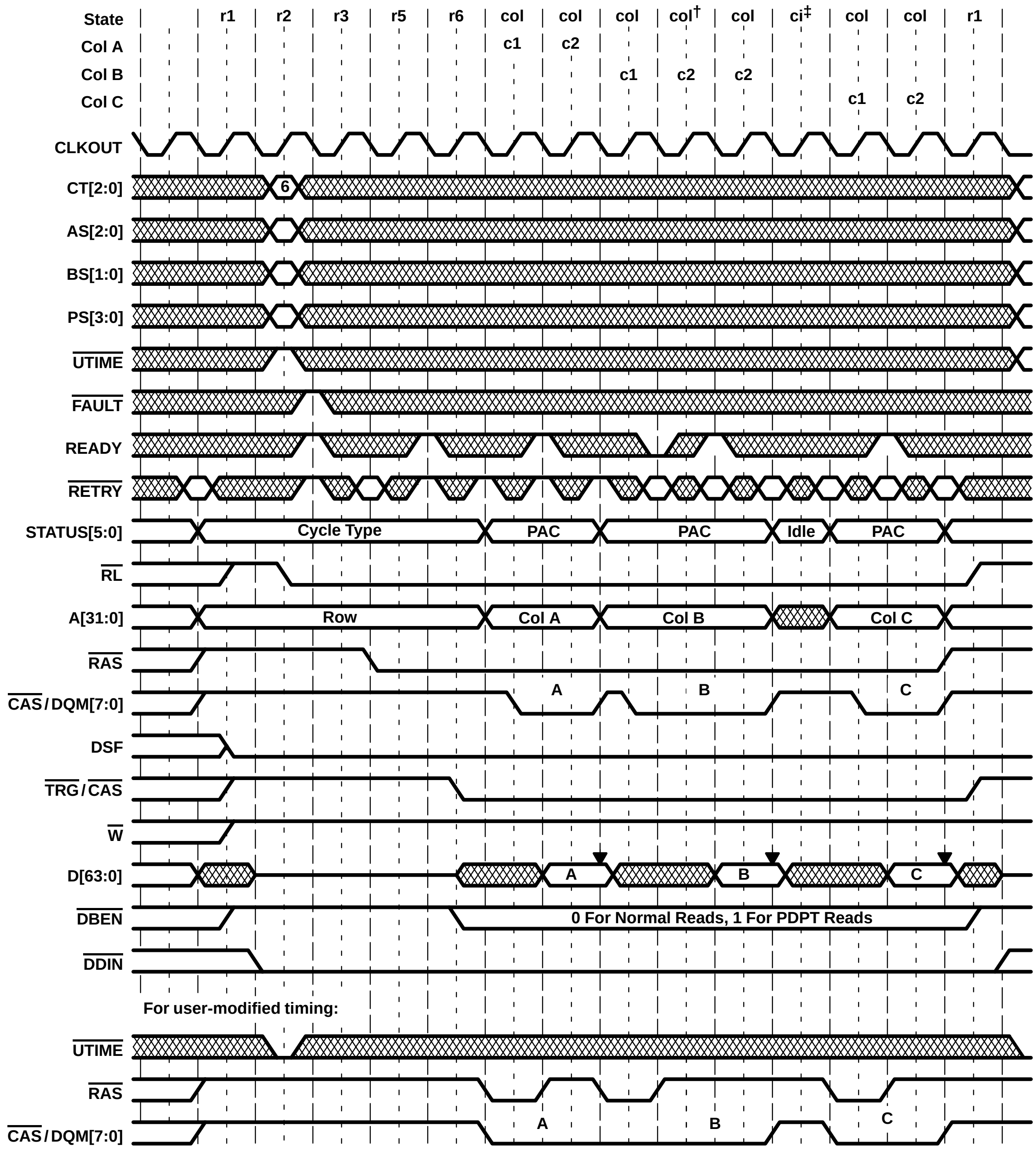


Figure 65. Nonpipelined 1 Cycle / Column Read-Cycle Timing

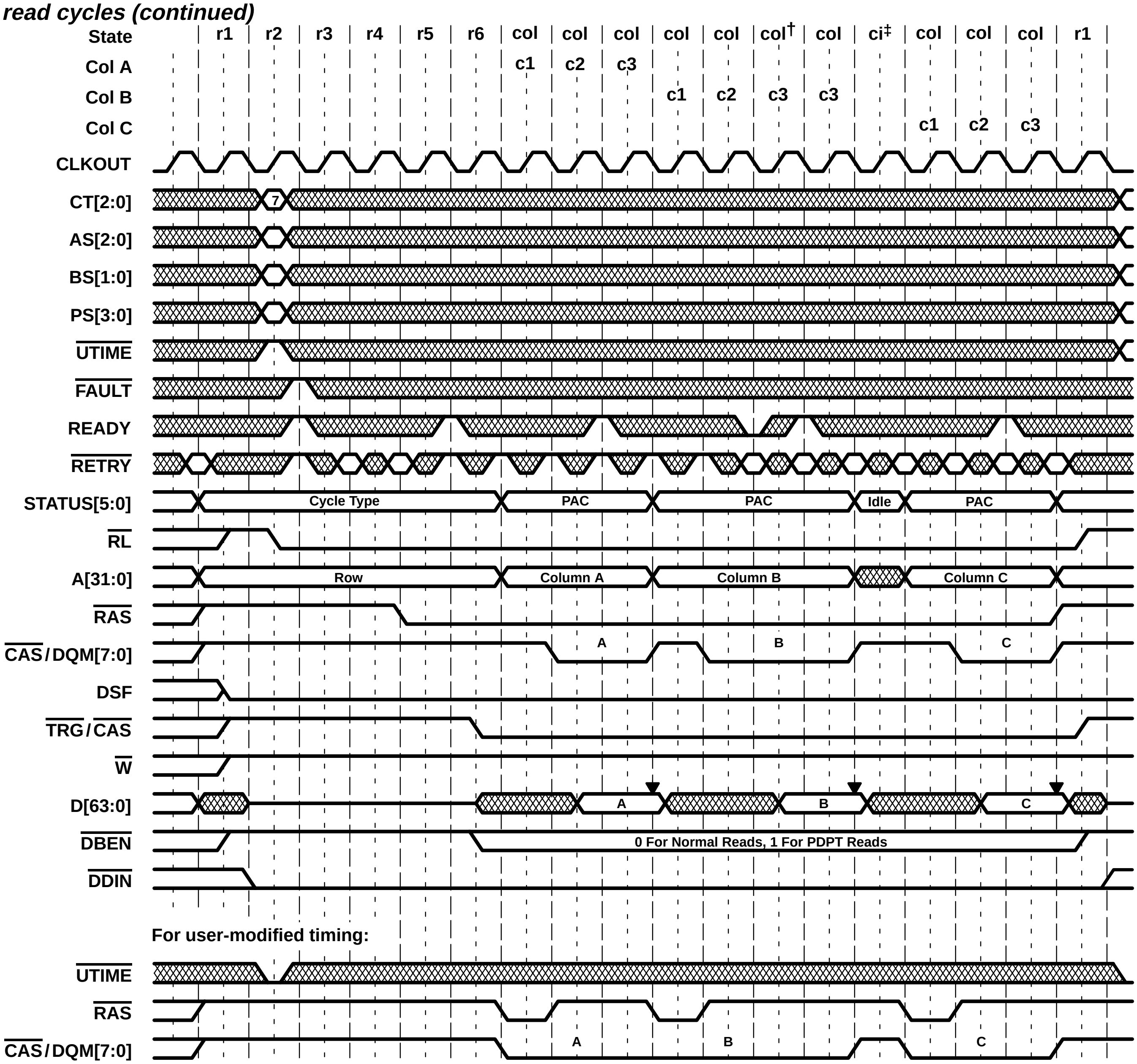
read cycles (continued)



† Wait state inserted by external logic (example)
‡ Internally generated pipeline bubble (example)

Figure 66. 2 Cycles / Column Read-Cycle Timing

read cycles (continued)



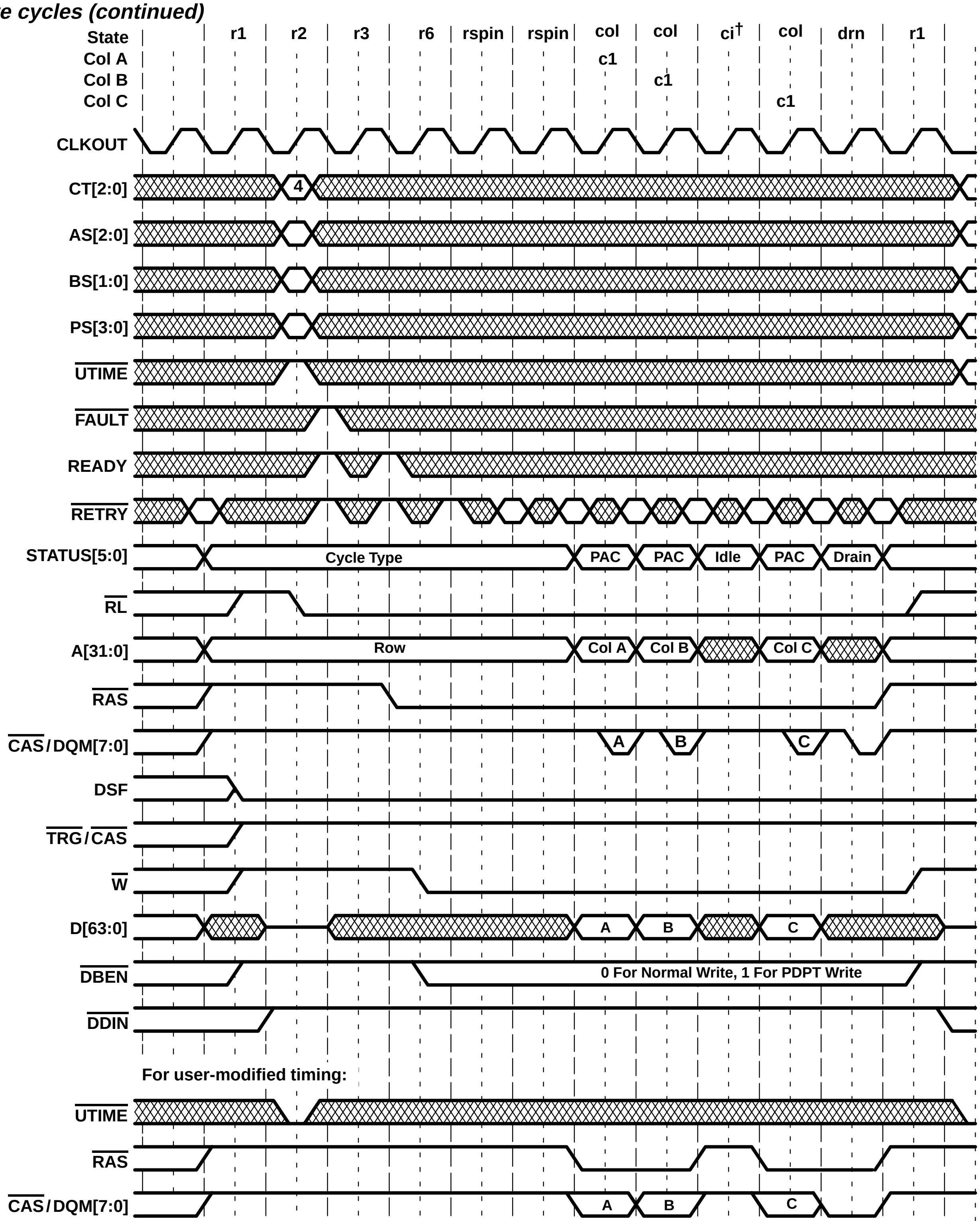
† Wait state inserted by external logic (example)
‡ Internally generated pipeline bubble (example)

Figure 67. 3 Cycles/Column Read-Cycle Timing

write cycles

Write cycles transfer data from the ’C80 to external memory. These cycles can occur as a result of a packet transfer, a DEA request, or an MP data cache write-back. During the cycle $\overline{\text{TRG}}/\overline{\text{CAS}}$ is held high, $\overline{\text{W}}$ is driven low after the fall of $\overline{\text{RAS}}$ to enable early-write cycles, and $\overline{\text{DDIN}}$ is high so that data transceivers drive toward memory. The TC drives data out on D[63:0] and indicates valid bytes by activating the appropriate $\overline{\text{CAS}}$/DQM strobes. During peripheral device packet transfers, $\overline{\text{DBEN}}$ remains high and D[63:0] is placed in high impedance so that the peripheral device can drive data into the memory. Write cycles are shown in Figure 68 through Figure 71.

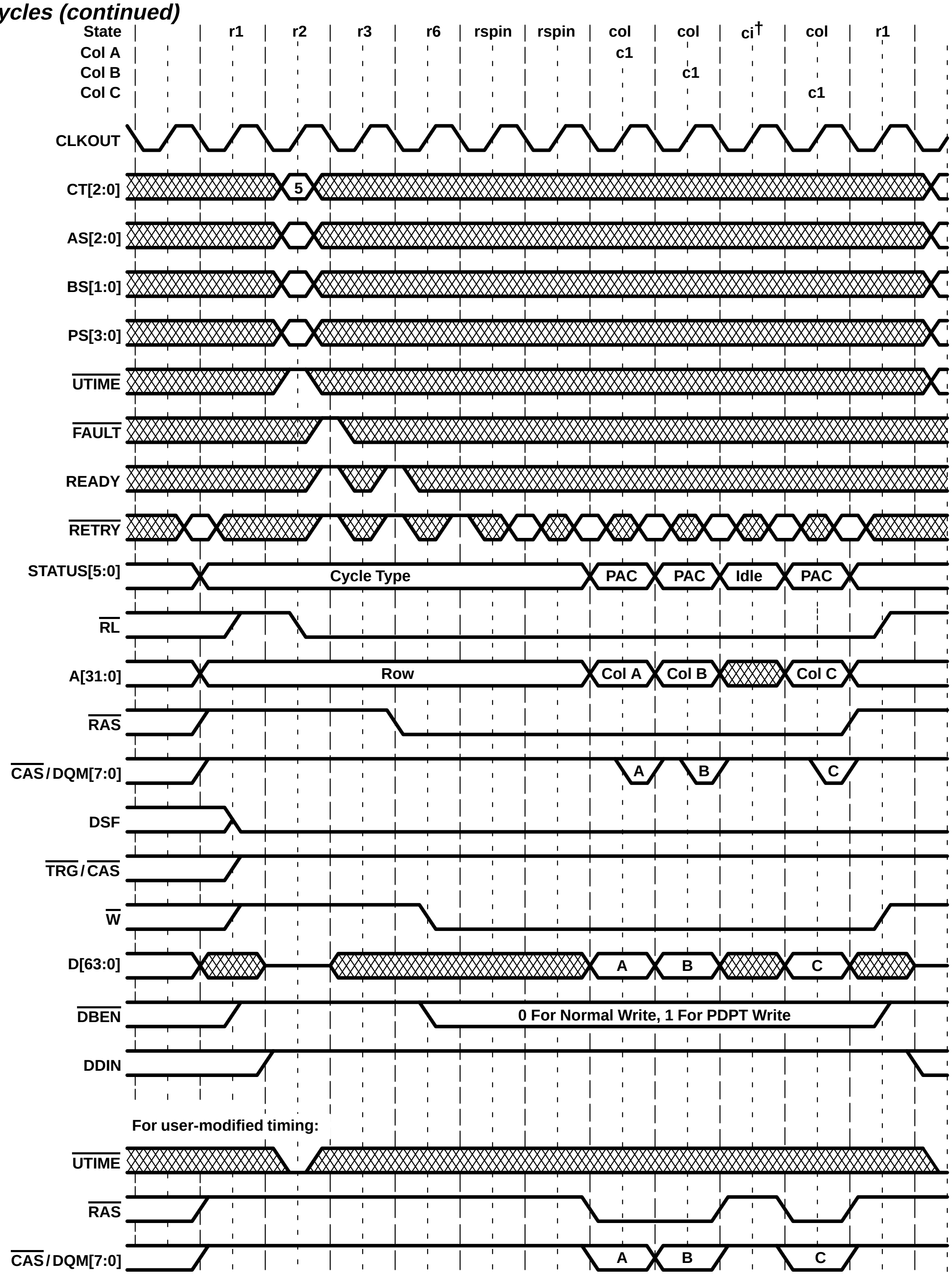
write cycles (continued)



† Internally generated pipeline bubble (example)

Figure 68. Pipelined 1 Cycle / Column Write-Cycle Timing

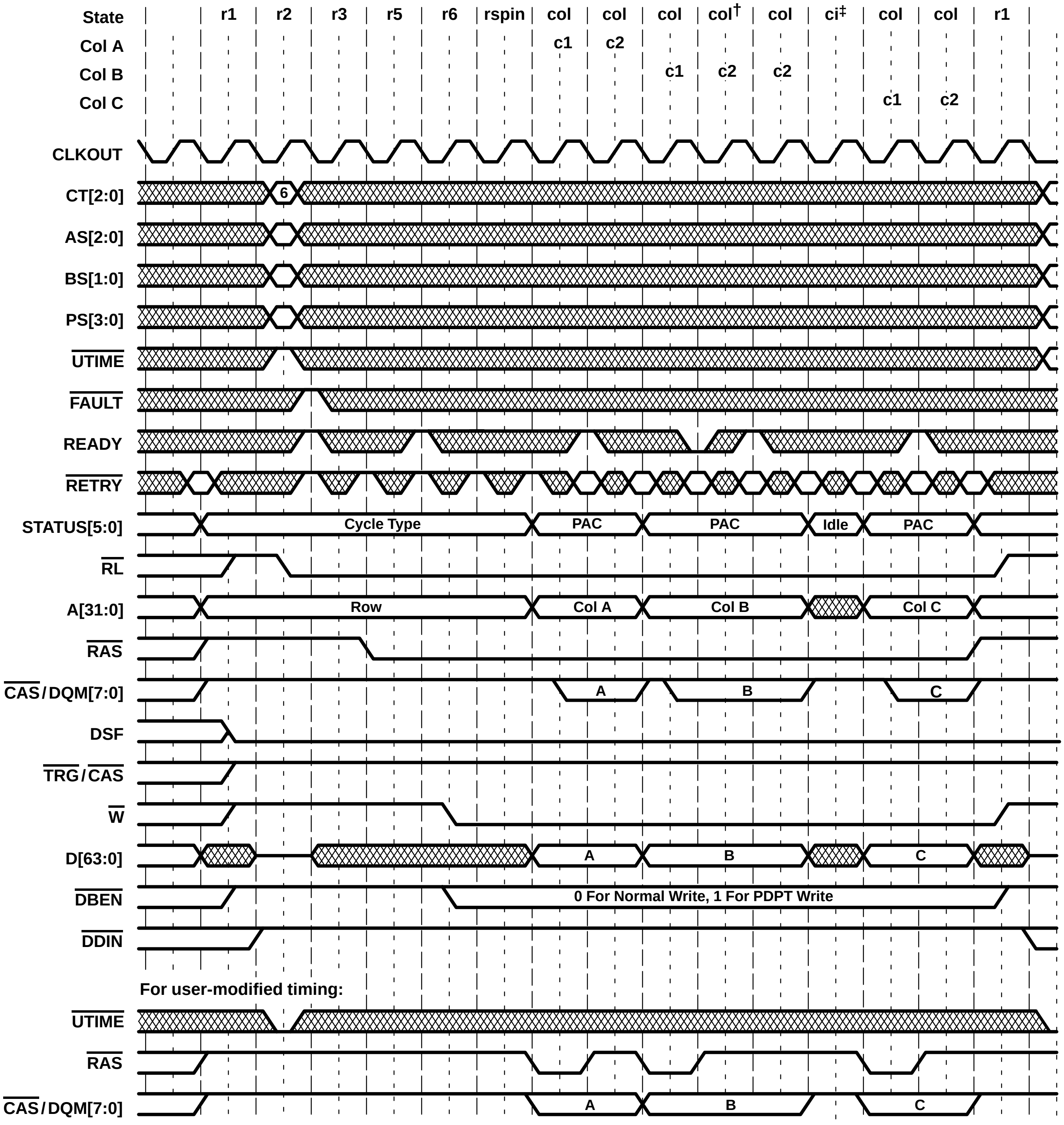
write cycles (continued)



† Internally generated pipeline bubble (example)

Figure 69. Nonpipelined 1 Cycle / Column Write-Cycle Timing

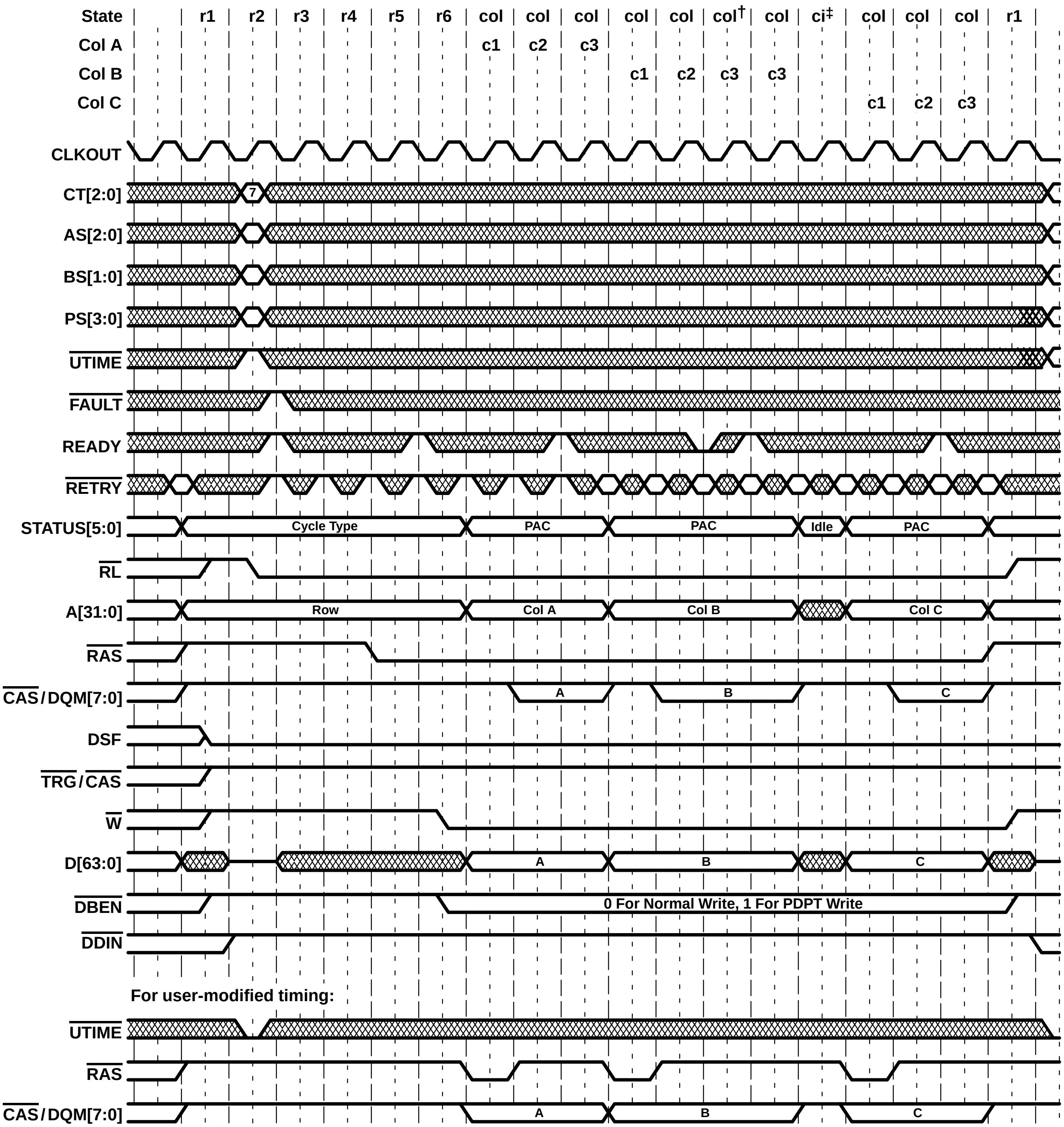
write cycles (continued)



† Wait state inserted by external logic (example)
‡ Internally generated pipeline bubble (example)

Figure 70. 2 Cycles / Column Write-Cycle Timing

write cycles (continued)



† Wait state inserted by external logic (example)
‡ Internally generated pipeline bubble (example)

Figure 71. 3 Cycles/Column Write-Cycle Timing

load-color-register cycles

Load-color-register (LCR) cycles are used to load a VRAM's color register prior to performing a block write. LCR cycles are supported only on 64-bit data buses. An LCR cycle closely resembles a normal write cycle because it writes into a VRAM. The difference is that the DSF output is high at both the fall of $\overline{\text{RAS}}$ and the fall of $\overline{\text{CAS}}$/DQM. Also, because the VRAM color register is a single location, only one column access occurs.

The row address that is output by the TC is used for bank decode only. Normally all VRAM banks should be selected during an LCR cycle because another LCR cycle will not occur when a block-write memory page change occurs. The column address that is output during an LCR is likewise irrelevant because the VRAM color register is the only location written. All $\overline{\text{CAS}}$/DQM strobes are active during an LCR cycle.

The $\overline{\text{RETRY}}$ input is sampled during LCR column states and must be valid high or low. Asserting $\overline{\text{RETRY}}$ at column time has no effect, however, because only one column access is performed.

If the BS[1:0] inputs indicate that the addressed memory supports only simulated block writes, the LCR cycle will be changed into a normal write cycle at the start of the simulated block write. Load color register cycles timing is shown in Figure 72 through Figure 75.

load-color-register cycles (continued)

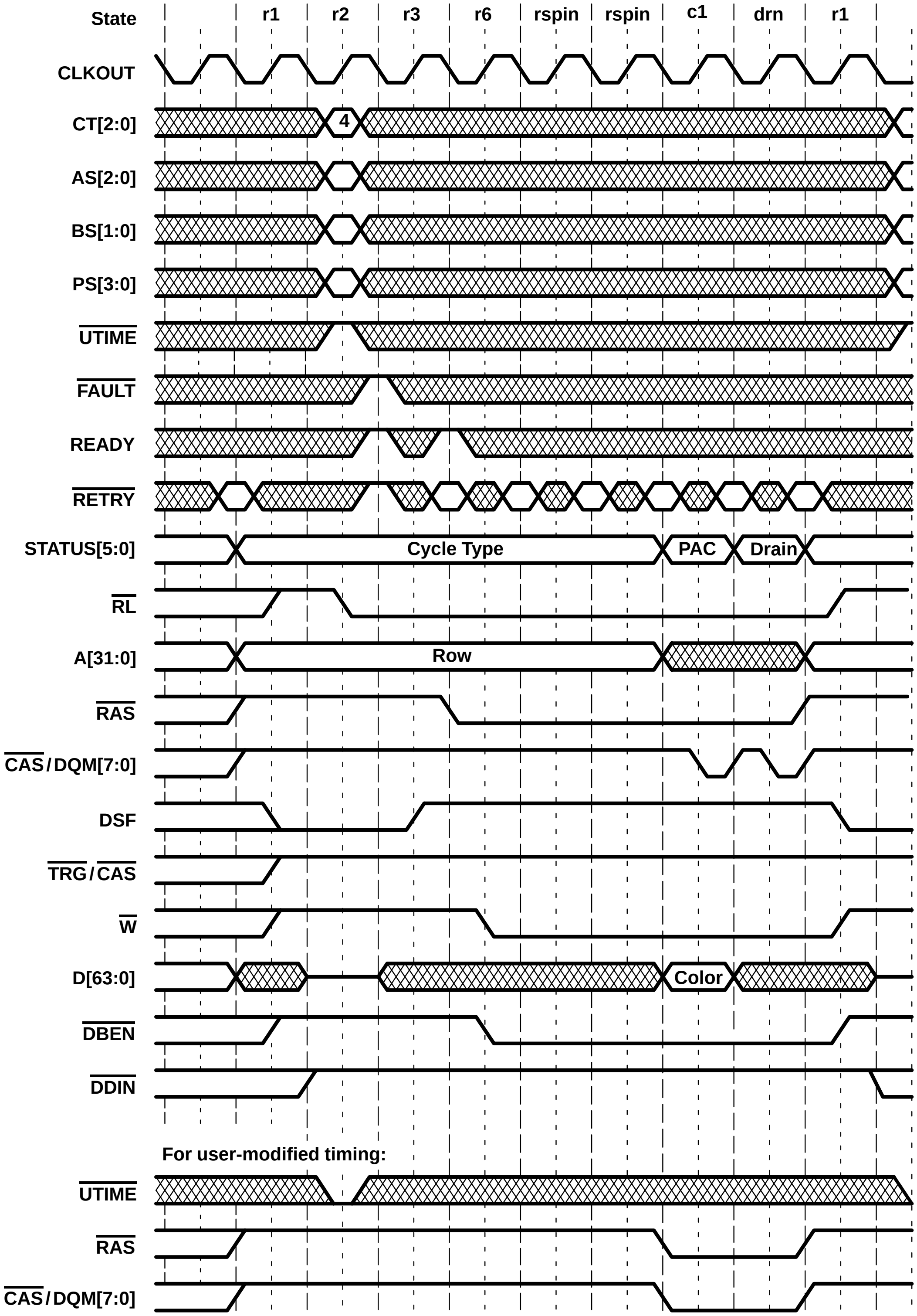


Figure 72. Pipelined 1 Cycle / Column Load-Color-Register-Cycle Timing

load-color-register cycles (continued)

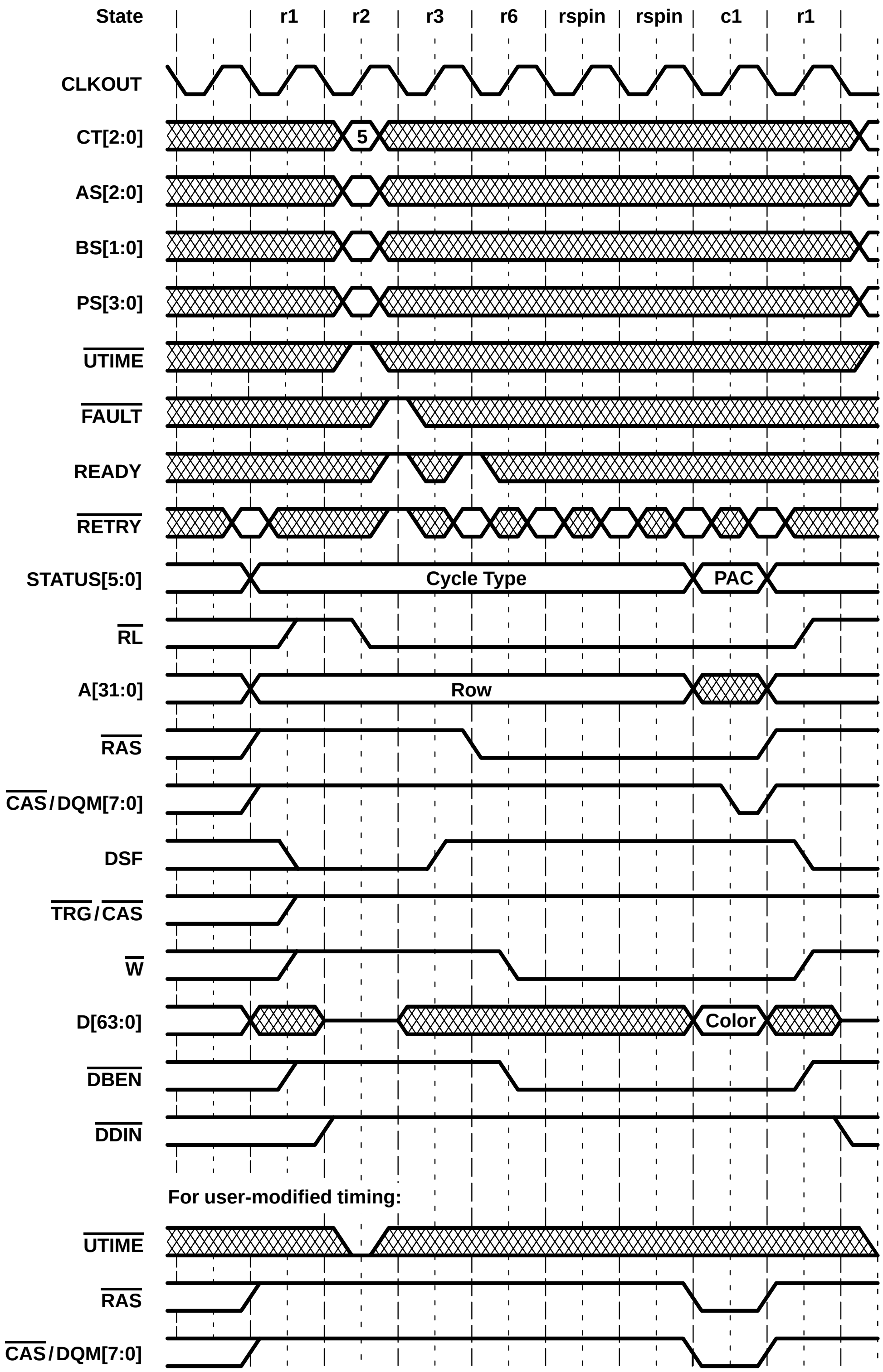


Figure 73. Nonpipelined 1 Cycle / Column Load-Color-Register-Cycle Timing

load-color-register cycles (continued)

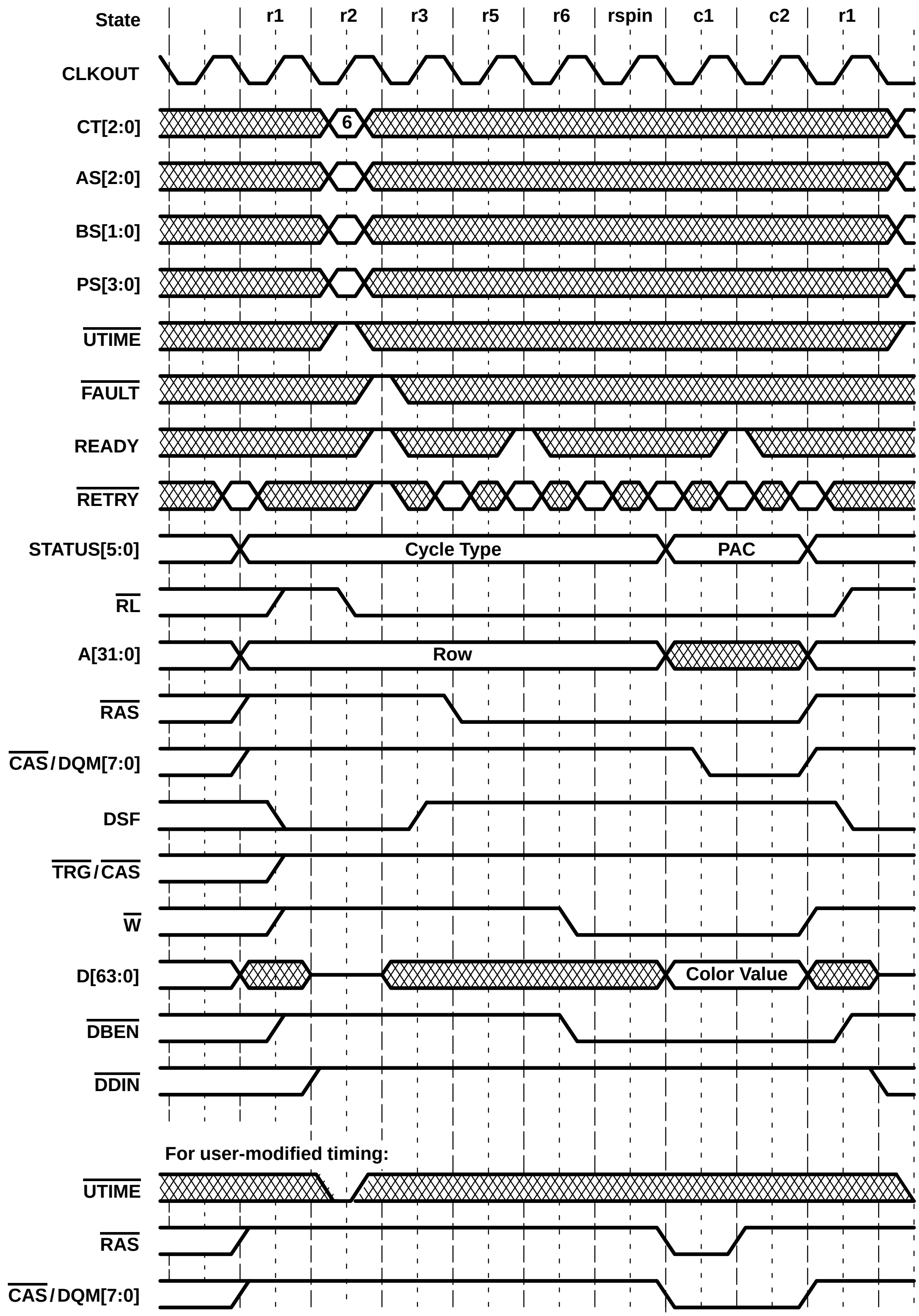


Figure 74. 2 Cycles/Column Load-Color-Register-Cycle Timing

load-color-register cycles (continued)

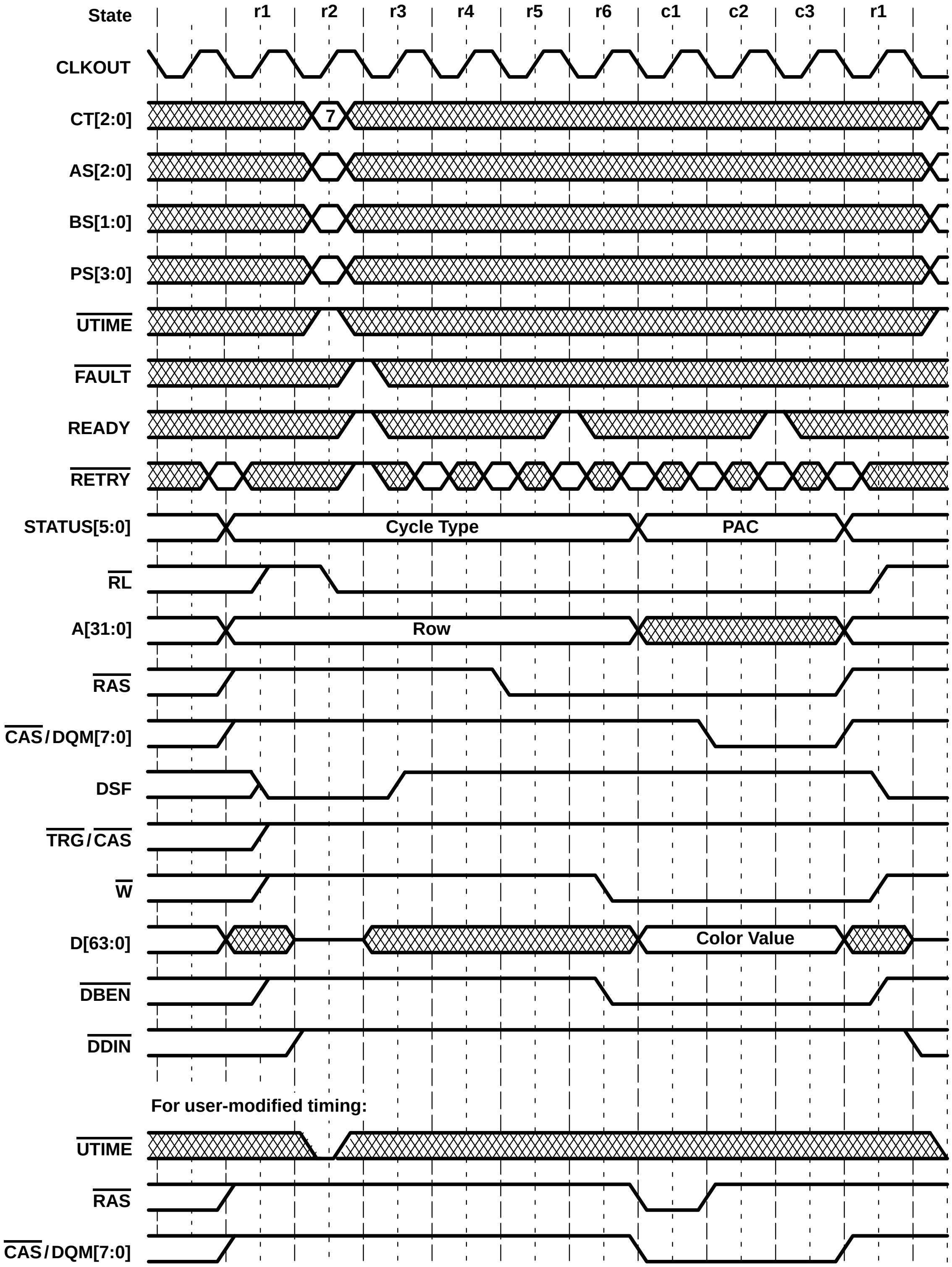


Figure 75. 3 Cycles/Column Load-Color-Register-Cycle Timing

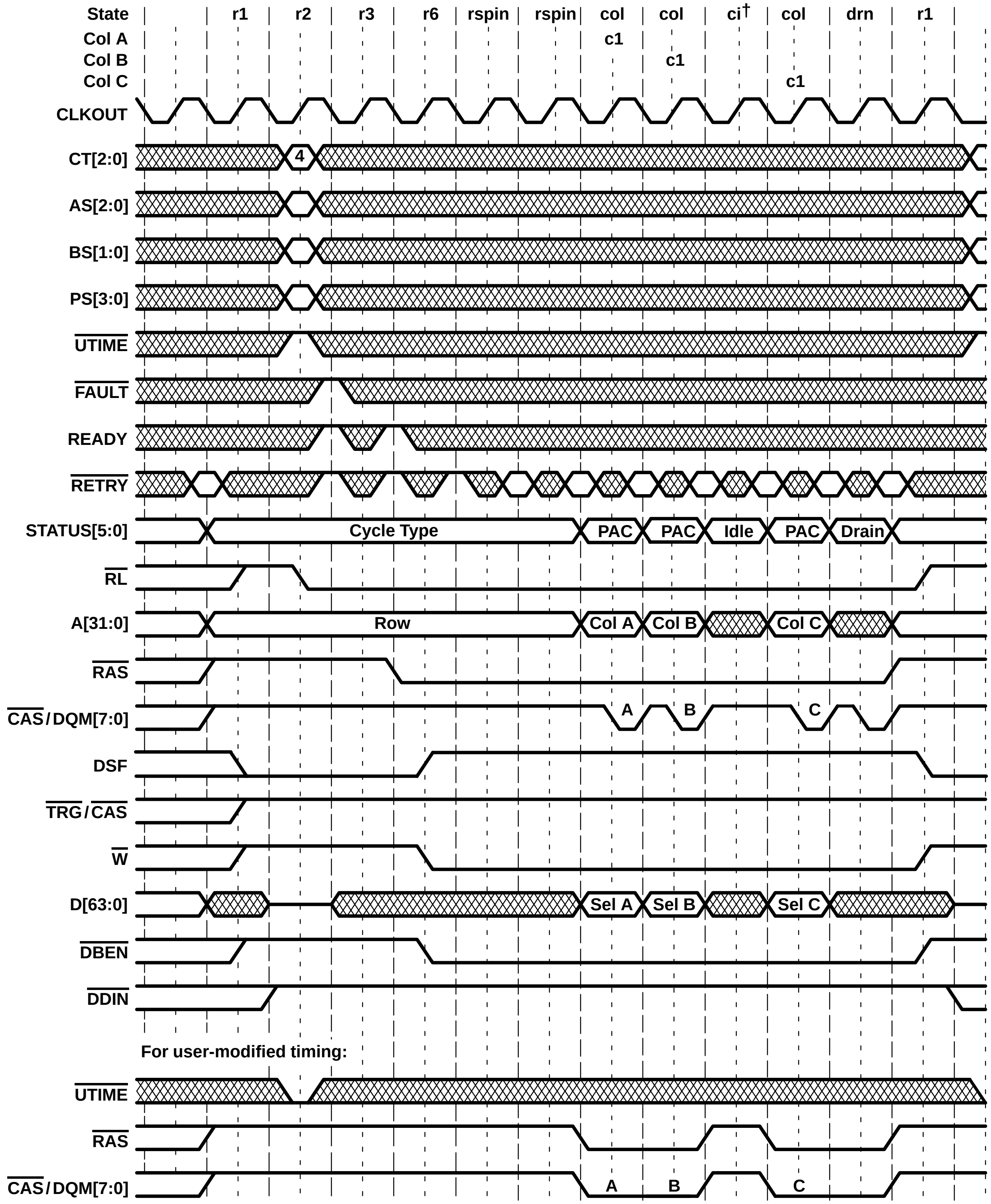
block-write cycles

Block-write cycles cause the data stored in the VRAM color registers to be written to the memory locations enabled by the appropriate data bits output on the D[63:0] bus. This allows up to a total of 64 bytes (depending on the type of block write being used) to be written in a single-column access. This cycle is identical to a standard write cycle with the following exceptions:

- DSF is active (high) at the fall of $\overline{\text{CAS}}$, enabling the block-write function within the VRAMs.
- Only 64-bit bus sizes are supported during block write; therefore, BS[1:0] inputs are used to indicate the type of block write that is supported by the addressed VRAMs, rather than the bus size.
- The two or three LSBs (depending on the type of block write) of the column address are ignored by the VRAMs because these column locations are specified by the data inputs.
- The values output by the TC on D[63:0] represent the column locations to be written to, using the color-register value. Depending on the type of block write supported by the VRAM, all of the data bits are not necessarily used by the VRAMs.
- Block writes always begin with a row access. Upon completion of a block write, the memory interface returns to state r1 to await the next access.

See Figure 76 through Figure 79 for block-write cycle timing.

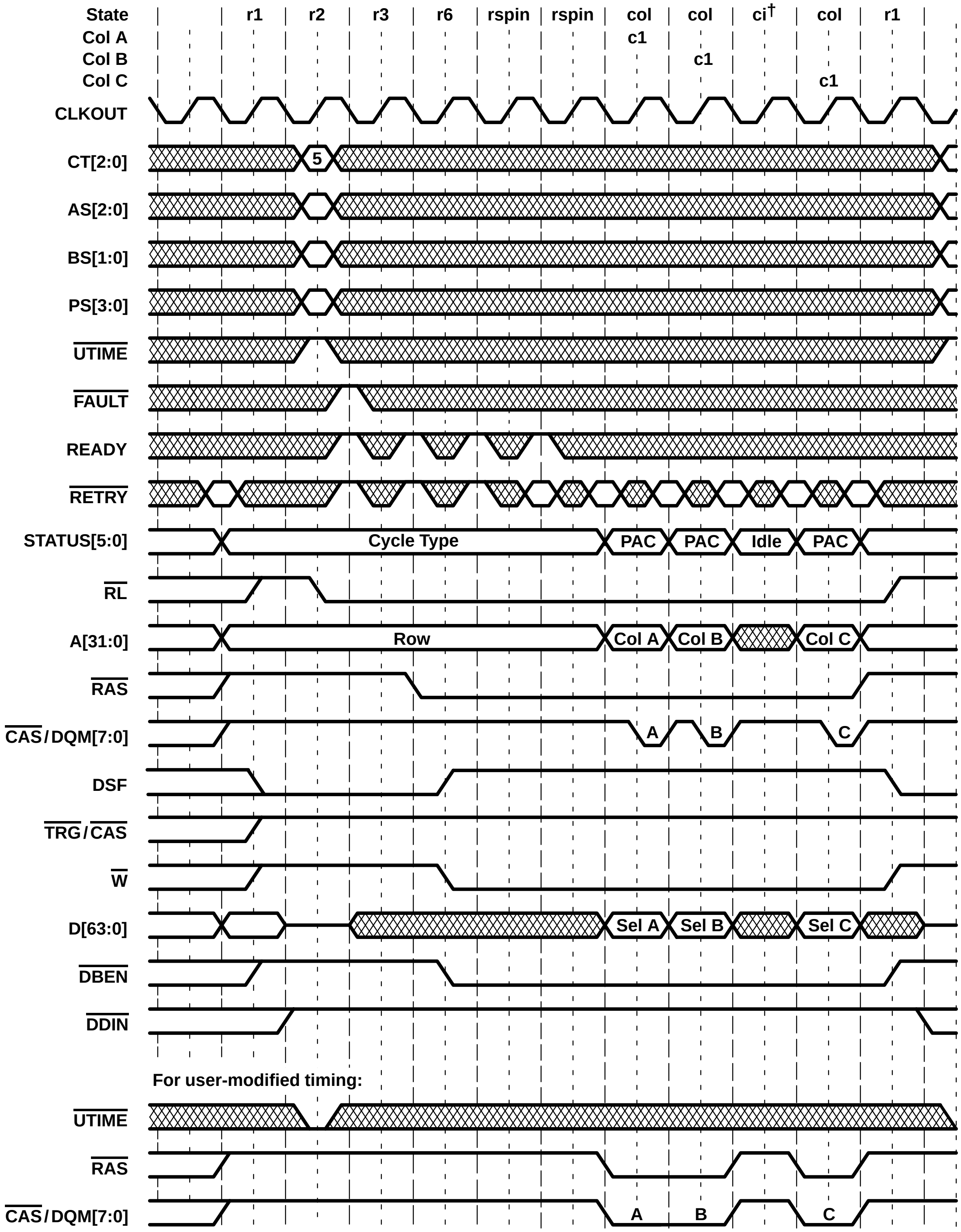
block-write cycles (continued)



† Internally generated pipline bubble (example)

Figure 76. Pipelined 1 Cycle/Column Block-Write-Cycle Timing

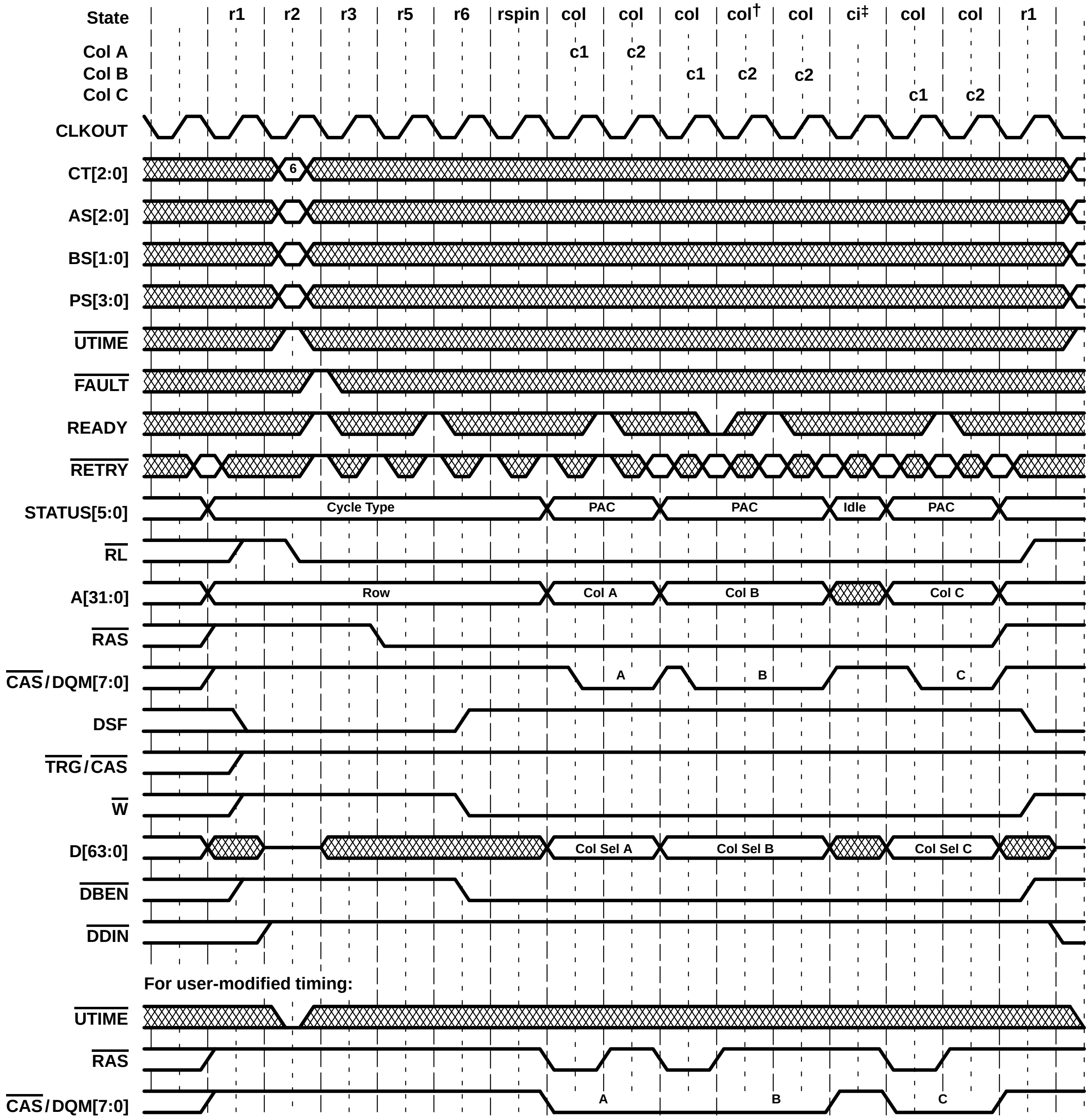
block-write cycles (continued)



† Internally generated pipline bubble (example)

Figure 77. Nonpipelined 1 Cycle/Column Block-Write-Cycle Timing

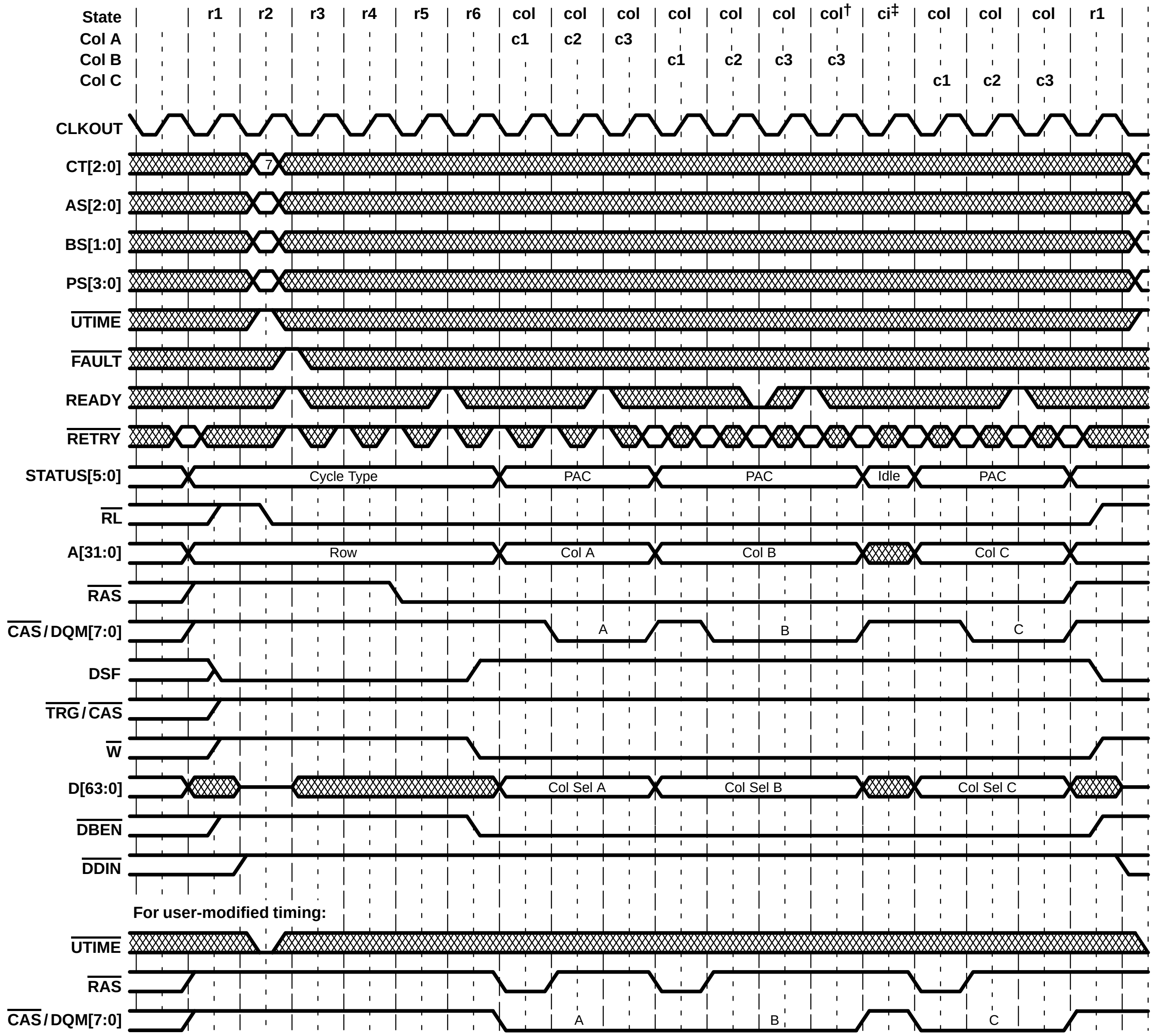
block-write cycles (continued)



† Wait state inserted by external logic (example)
‡ Internally generated pipeline bubble (example)

Figure 78. 2 Cycles / Column Block-Write-Cycle Timing

block-write cycles (continued)



† Wait state inserted by external logic (example)
‡ Internally generated pipeline bubble (example)

Figure 79. 3 Cycles / Column Block-Write-Cycle Timing

transfer cycles

Read-transfer (memory-to-register) cycles transfer a row from the VRAM memory array into the VRAM shift register (SAM). This causes the entire SAM (both halves of the split SAM) to be loaded with the array data.

Split-register read-transfer (memory-to-split-register) cycles also transfer data from a row in the memory array to the SAM. However, these transfers cause only half of the SAM to be written. Split-register read transfers allow the inactive half of the SAM to be loaded with the new data while the other active half continues to shift data in or out.

Write-transfer (register-to-memory) cycles transfer data from the SAM into a row of the VRAM array. This transfer causes the entire SAM (both halves of the split SAM) to be written into the array.

Split-register write-transfer (split-register-to-memory) cycles also transfer data from the SAM to a row in the memory array. However, these transfers write only half of the SAM into the array. Split-register write transfers allow the inactive half of the SAM to be transferred into memory while the other (active) half continues to shift serial data in or out.

Read and split-read transfers resemble a standard read cycle. Write and split-write transfers resemble a standard write cycle. The $\overline{\text{TRG}}/\overline{\text{CAS}}$ output is driven low prior to the fall of $\overline{\text{RAS}}$ to indicate a transfer cycle. Only a single column access is performed so $\overline{\text{RETRY}}$, while required to be at a valid level, has no effect if asserted at column time. The value output on A[31:0] at column time represents the SAM tap point (see Figure 80 through Figure 86 for transfer cycle timing.

transfer cycles (continued)

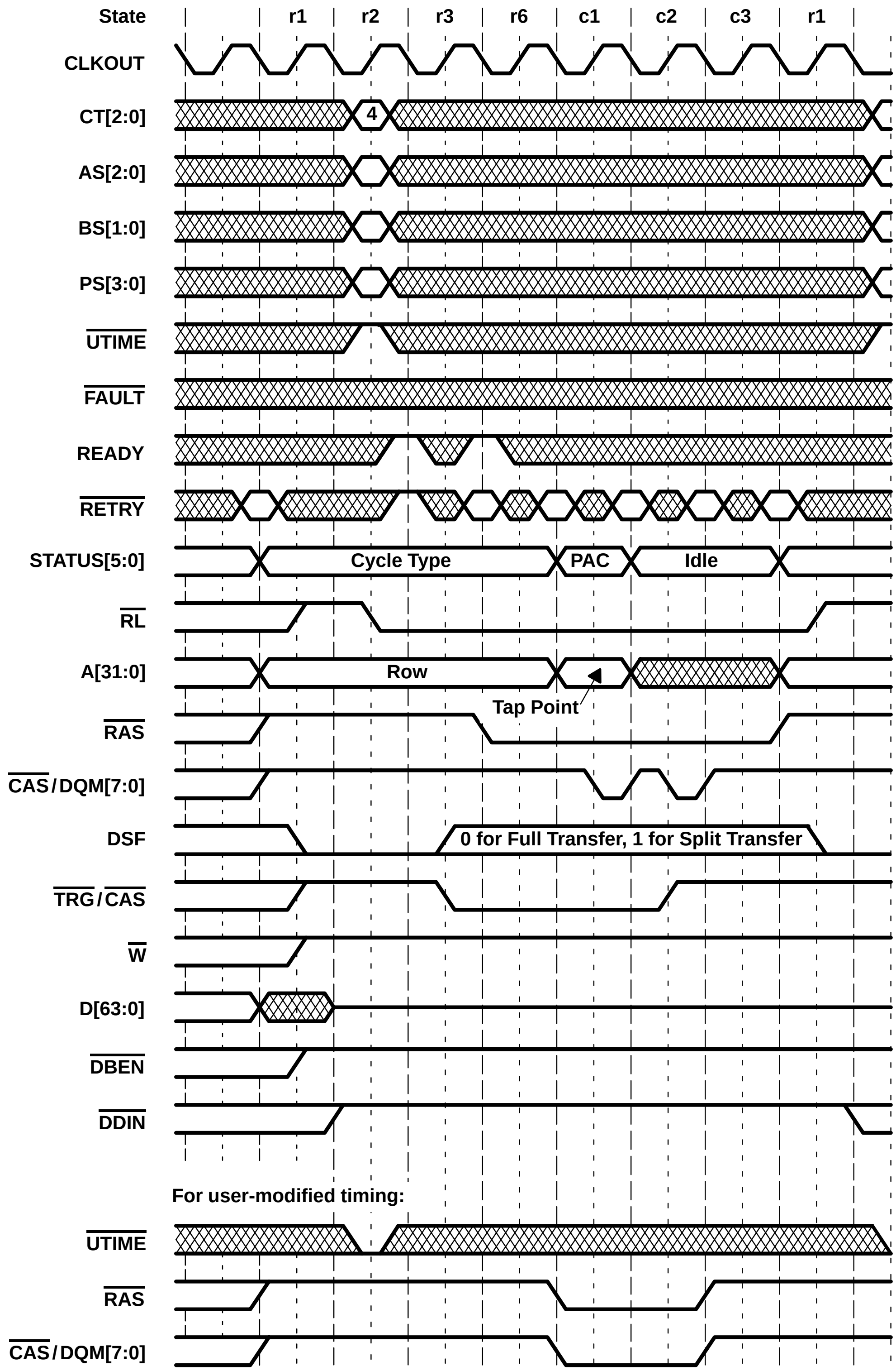


Figure 80. Pipelined 1 Cycle/Column Read-Transfer and Split-Register Read-Transfer-Cycle Timing

transfer cycles (continued)

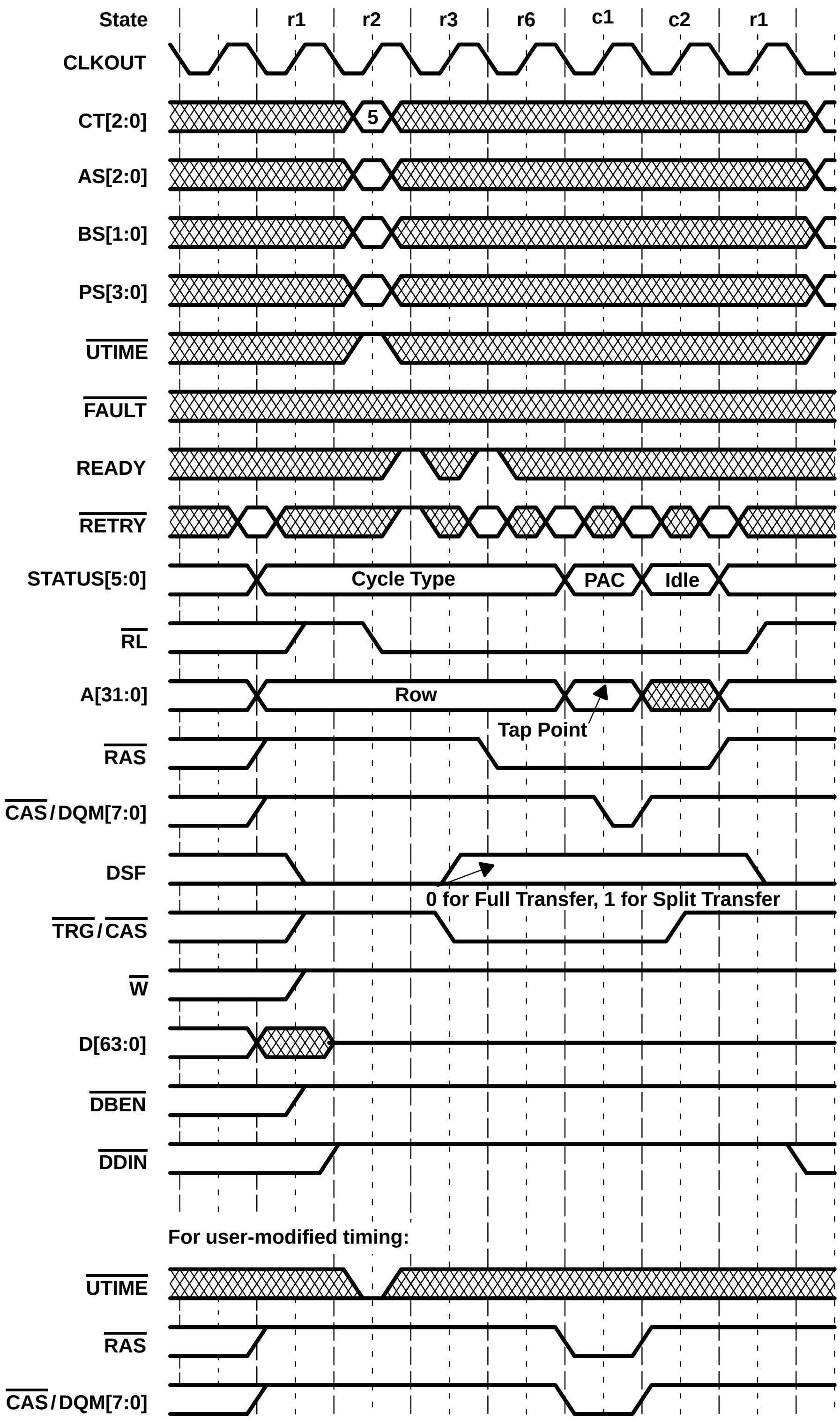


Figure 81. Nonpipelined 1 Cycle/Column Read-Transfer and Split-Register Read-Transfer-Cycle Timing

transfer cycles (continued)

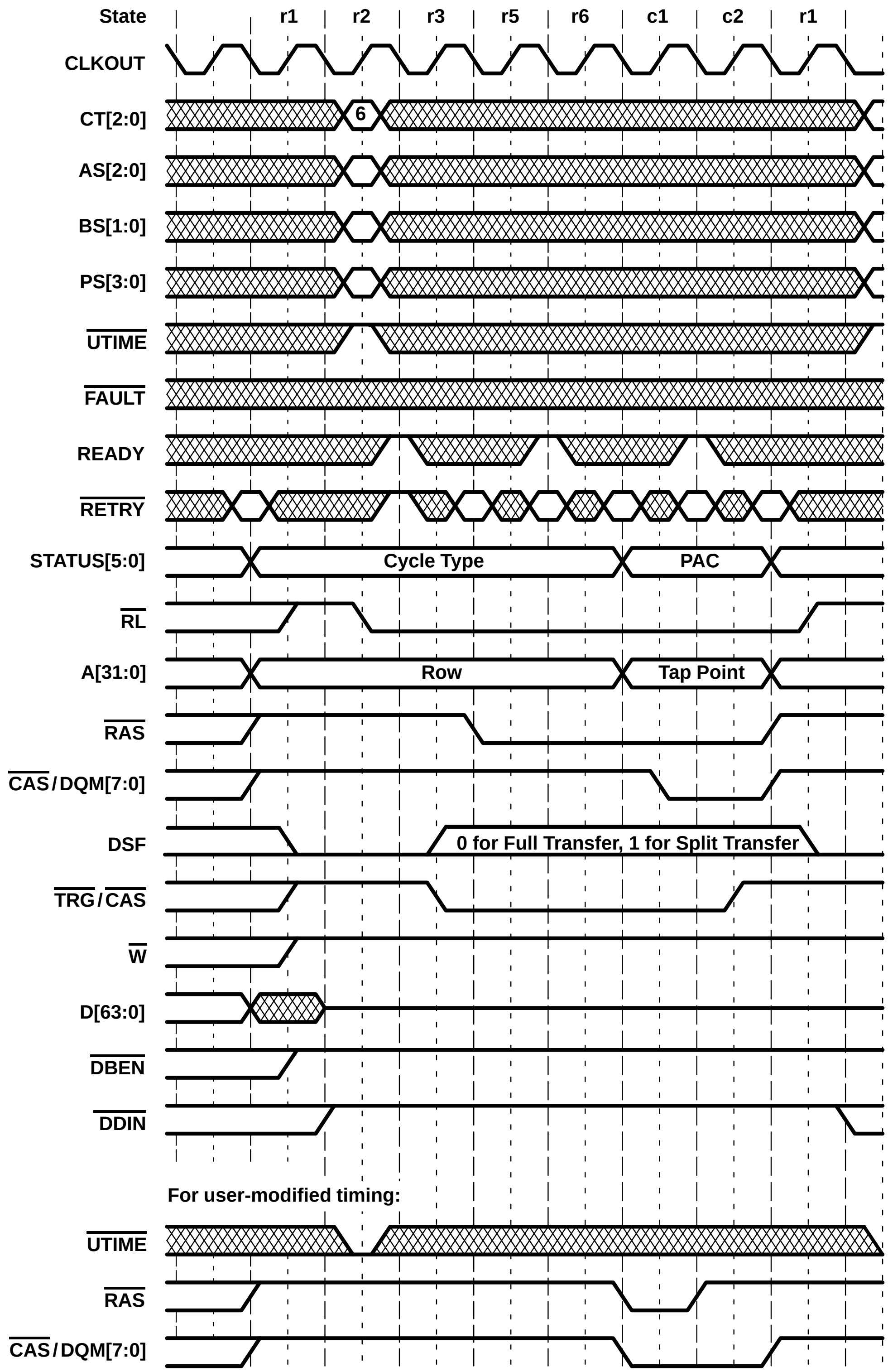


Figure 82. 2 Cycles/Column Read-Transfer and Split-Register Read-Transfer-Cycle Timing

transfer cycles (continued)

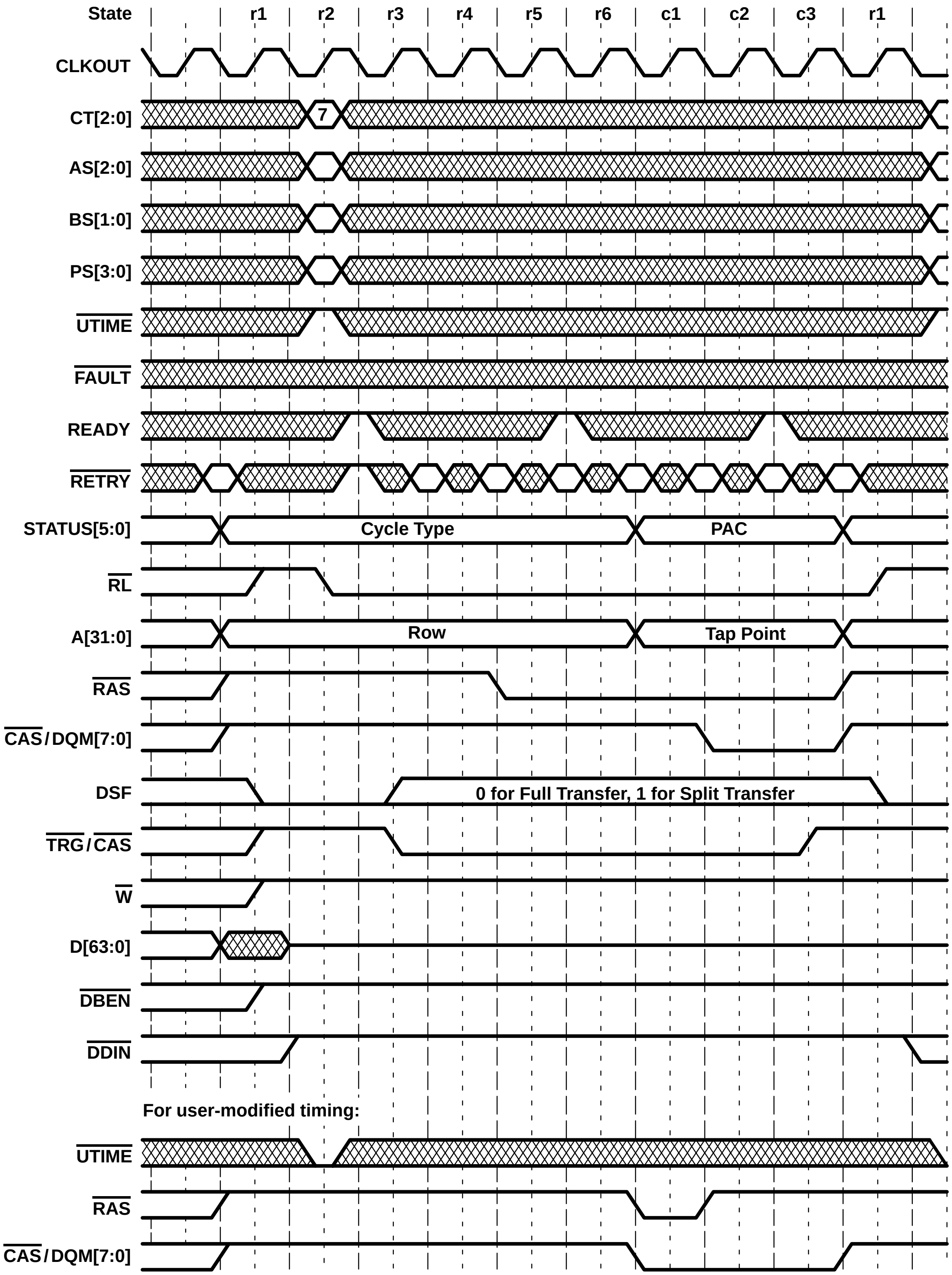


Figure 83. 3 Cycles/Column Read-Transfer and Split-Register Read-Transfer-Cycle Timing

transfer cycles (continued)

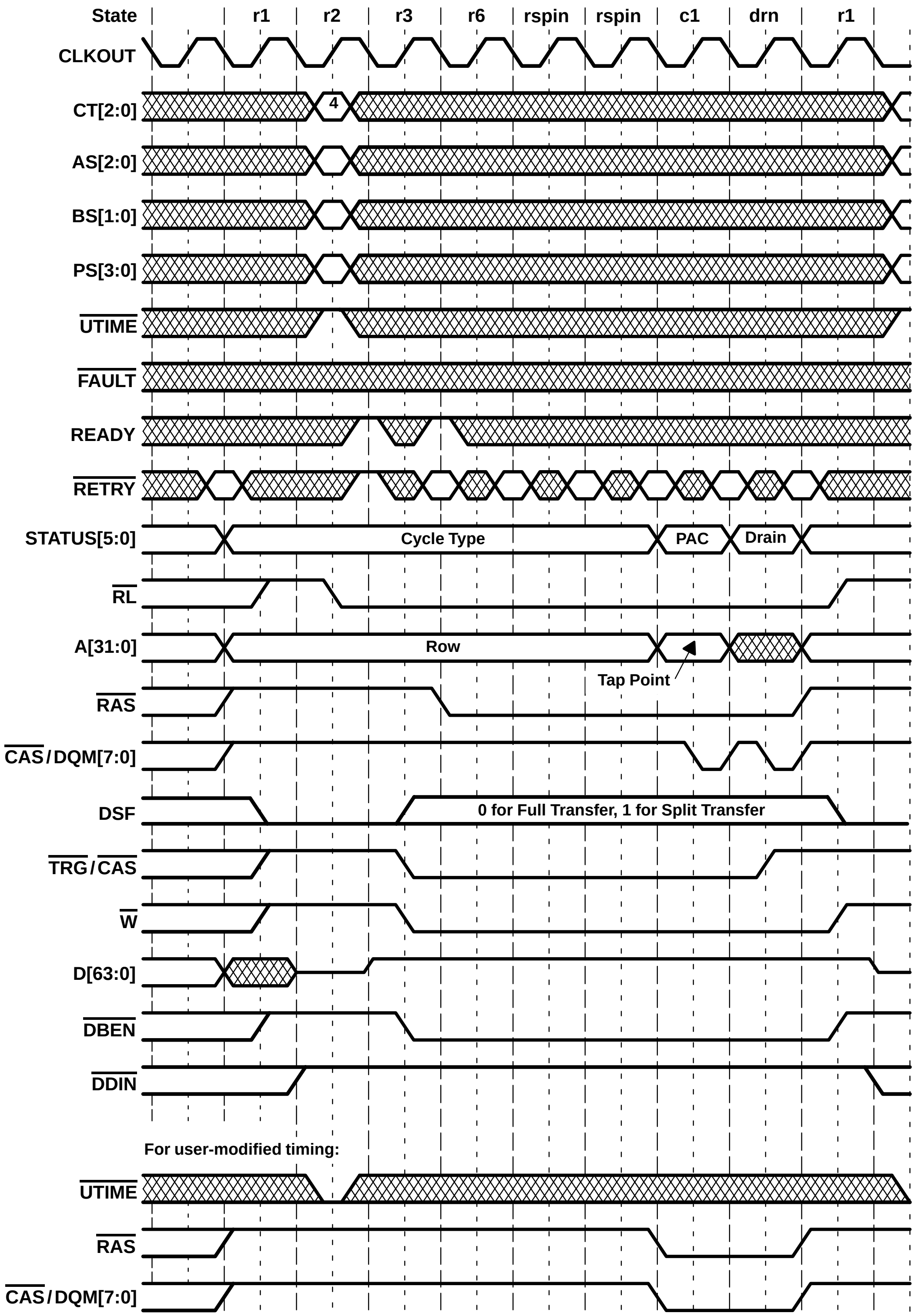


Figure 84. Pipelined 1 Cycle/Column Write-Transfer and Split-Register Write-Transfer-Cycle Timing

transfer cycles (continued)

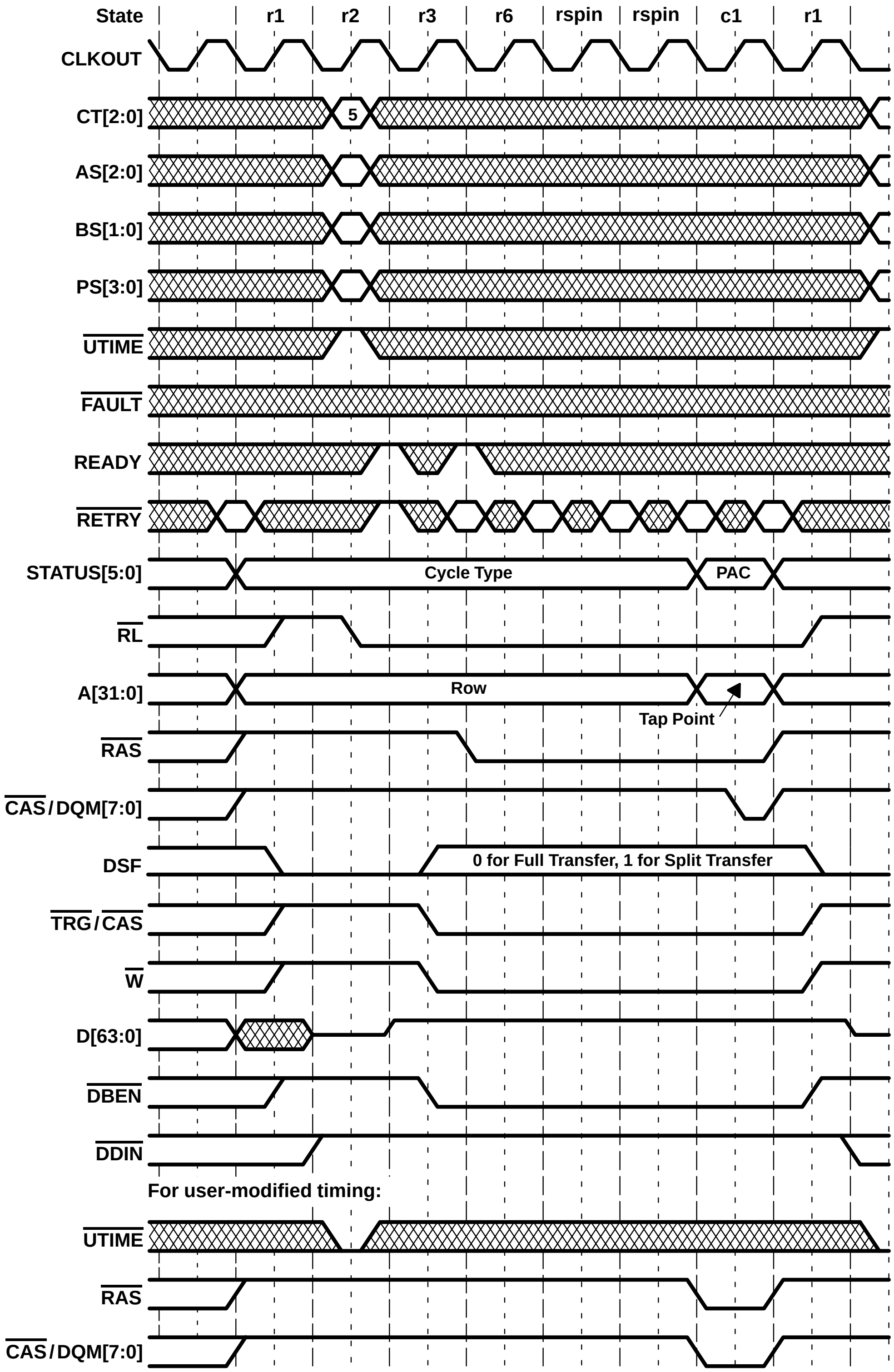


Figure 85. Nonpipelined 1 Cycle/Column Write-Transfer and Split-Register Write-Transfer-Cycle Timing

transfer cycles (continued)

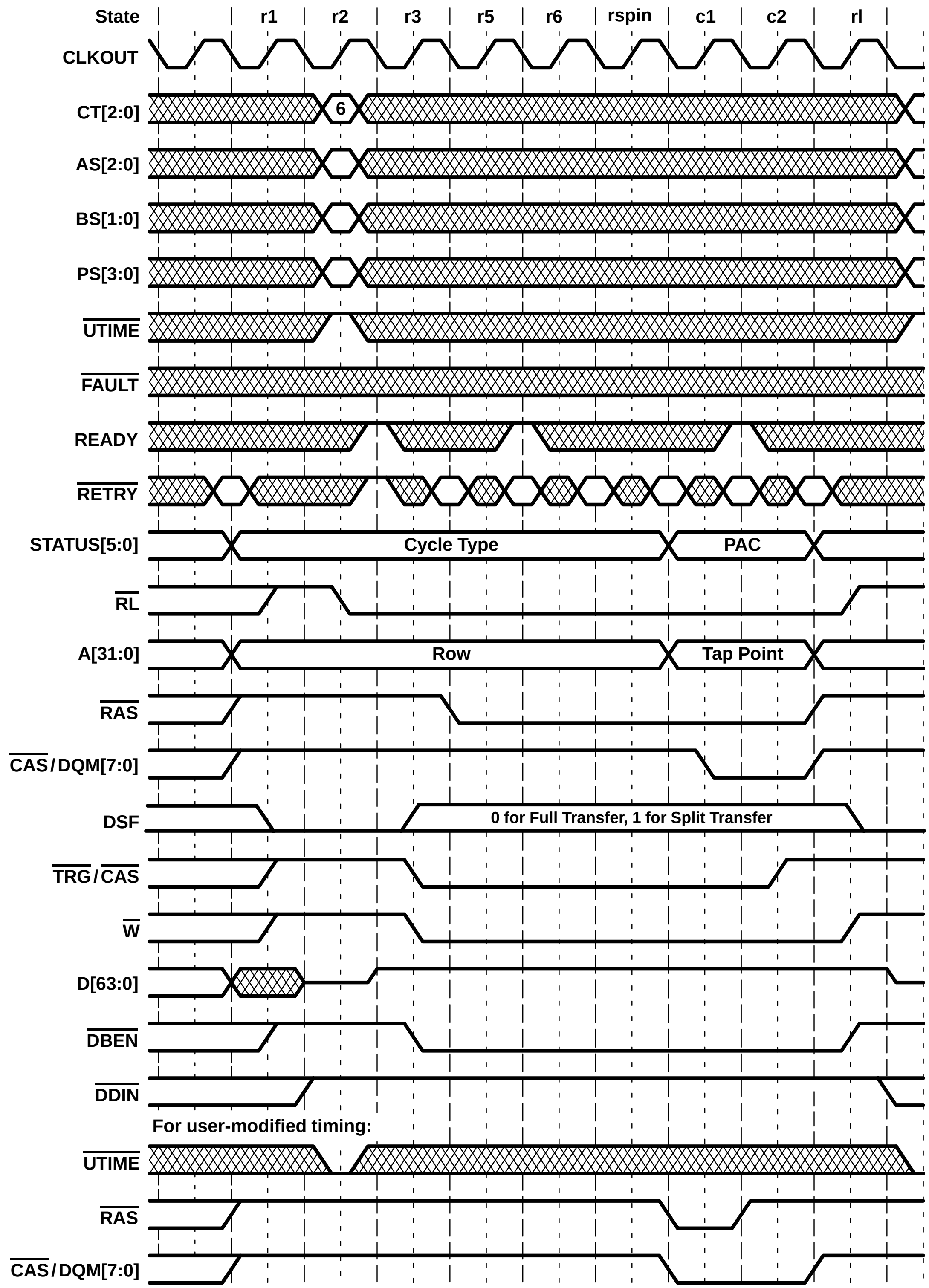


Figure 86. 2 Cycles/Column Write-Transfer and Split-Register Write-Transfer-Cycle Timing

transfer cycles (continued)

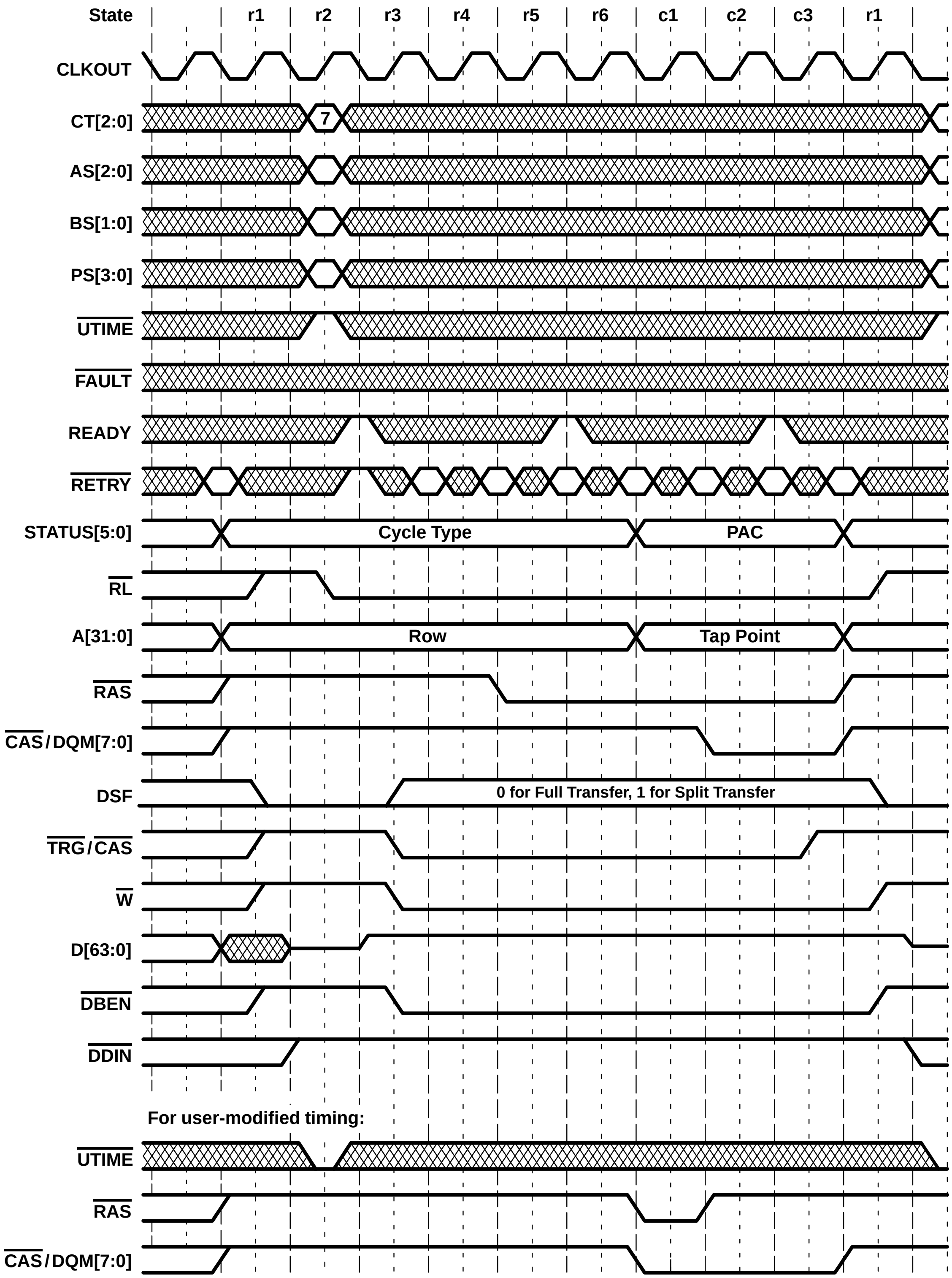


Figure 87. 3 Cycles/Column Write-Transfer and Split-Register Write-Transfer-Cycle Timing

refresh cycles

Refresh cycles are generated by the TC at the programmed refresh interval. They are characterized by the following signal activity:

- $\overline{\text{CAS}}$ falls prior to $\overline{\text{RAS}}$.
- All $\overline{\text{CAS}}$ pins ($\overline{\text{CAS}}$/DQM[7:0]) are active.
- $\overline{\text{TRG}}$, $\overline{\text{W}}$, and $\overline{\text{DBEN}}$ all remain inactive (high) because no data transfer occurs.
- DSF remains inactive (low).
- The data bus is driven to the high-impedance state.
- The upper half of the address bus (A[31:16]) contains the refresh pseudo-address and the lower half (A[15:0]) is driven to all zeros.
- If $\overline{\text{RETRY}}$ is asserted at any sample point during the cycle, the cycle timing is not modified. Instead, the pseudo-address and backlog counters are simply not decremented.
- Selecting user-modified timing has no effect on the cycles.
- Upon completion of the refresh cycle, the memory interface returns to state r1 to await the next access.

See Figure 88 through Figure 90 for refresh cycle timing.

refresh cycles (continued)

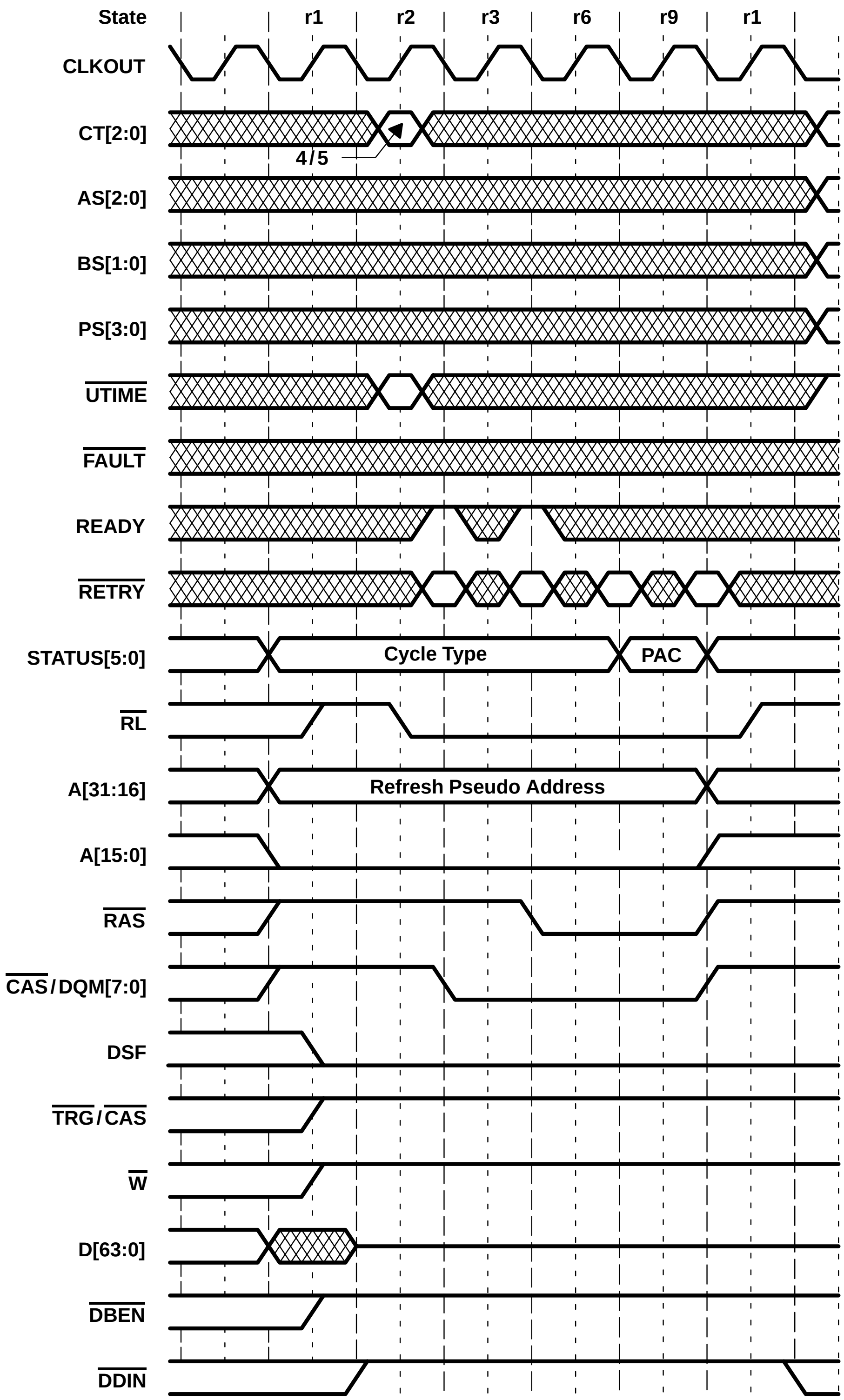


Figure 88. 1-Cycle/Column Refresh-Cycle Timing

refresh cycles (continued)

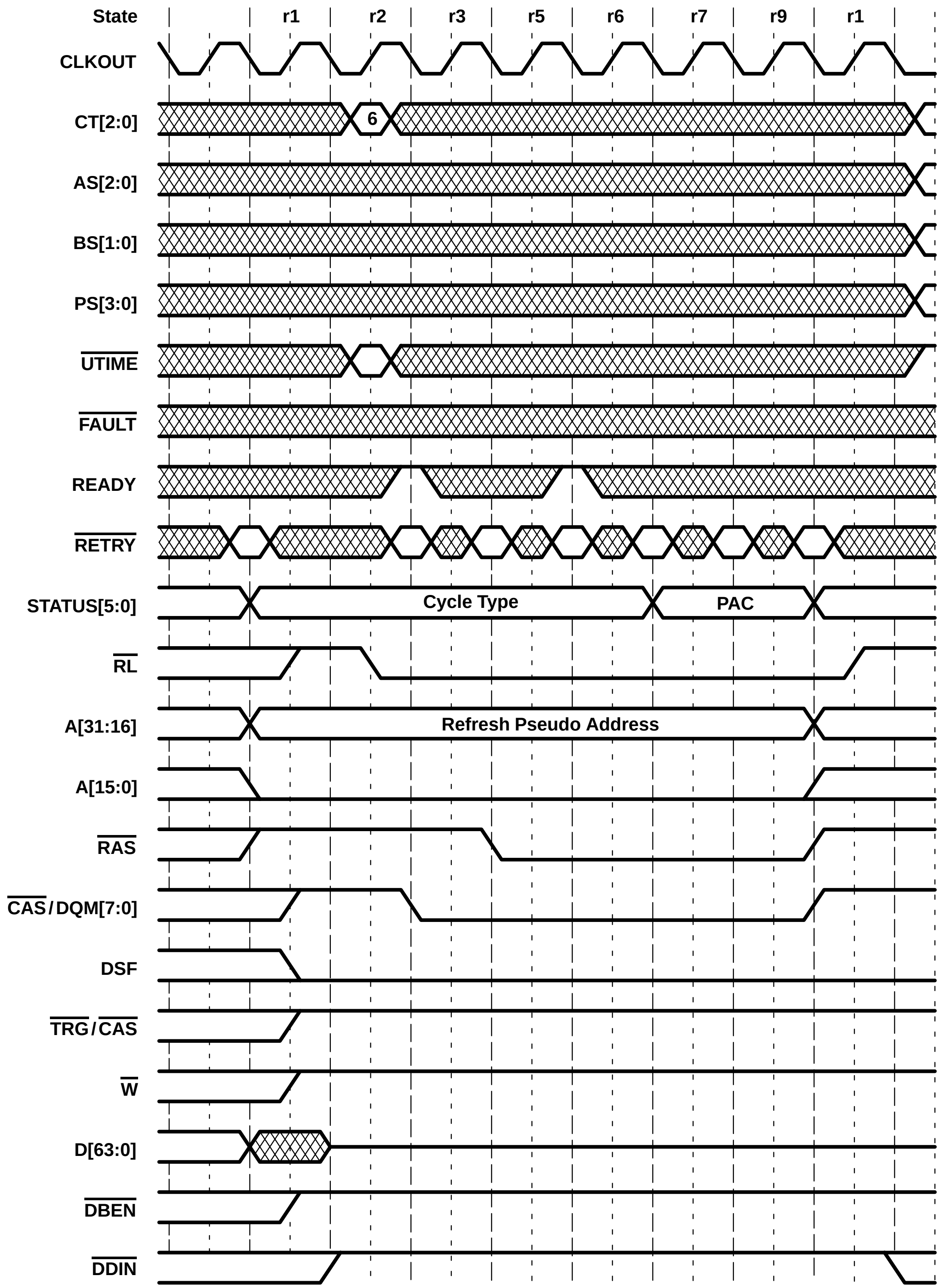


Figure 89. 2 Cycles/Column Refresh-Cycle Timing

refresh cycles (continued)

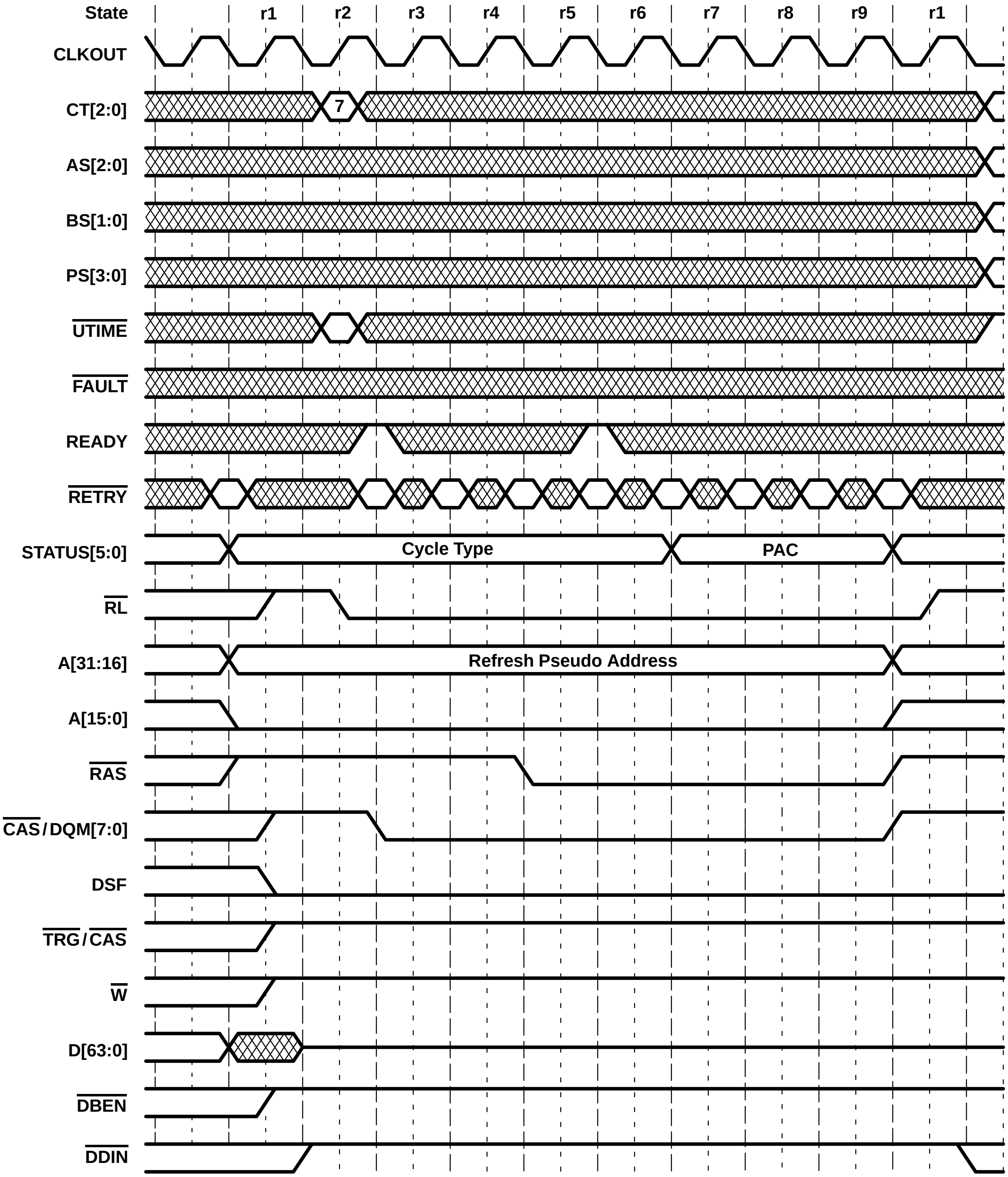


Figure 90. 3 Cycles/Column Refresh-Cycle Timing

SDRAM type cycles

The SDRAM type cycles support the use of SDRAM, SGRAM, or SVRAM devices for single-cycle memory accesses. While SDRAM cycles use the same state sequences as DRAM cycles, the memory-control signal transitions are modified to perform SDRAM command cycles. The supported SDRAM commands are:

DCAB	Deactivate (precharge) all banks
ACTV	Activate the selected bank and select the row
READ	Input starting column address and start read operation
WRT	Input starting column address and start write operation
MRS	Set SDRAM mode register
REFR	Auto-refresh cycle with internal address
SRS	Set special register (color register)
BLW	Block write

SDRAM cycles begin with an activate (ACTV) command followed by the requested column accesses. When a memory-page change occurs, the selected bank is deactivated with a DCAB command.

The TMS320C80 supports read latencies of 2, 3, or 4 cycles and burst lengths of 1 or 2. These are selected by the CT code and $\overline{\text{UTIME}}$ value input at the start of the access.

The column pipelines for SDRAM accesses are shown in Figure 91. Idle cycles can occur after necessary column accesses have completed or between column accesses due to “bubbles” in the TC data flow pipeline. The pipeline diagrams show the pipeline stages for each access type and when the $\overline{\text{CAS}}$/DQM signal corresponding to the column access is activated.

SDRAM type cycles (continued)

$\overline{CAS}$/DQM	A	B	C			
Col A	c1	c2	c3			
Col B		c1	c2	c3		
Col C			c1	c2	c3	
Idle				ci	ci	ci

Burst-length 1, 2 cycle latency reads, read transfers, split-read transfers

$\overline{CAS}$/DQM		A	B	C			
Col A	c1	c2	c3	c4			
Col B		c1	c2	c3	c4		
Col C			c1	c2	c3	c4	
Idle				ci	ci	ci	ci

Burst-length 1, 3 cycle latency reads, read transfers, split-read transfers

$\overline{CAS}$/DQM		A	B	C				
Col A	c1	c2	c3	c4	c5			
Col B		c1	c2	c3	c4	c5		
Col C			c1	c2	c3	c4	c5	
Idle				ci	ci	ci	ci	ci

Burst-length 1, 4 cycle latency reads, read transfers,split-read transfers.

$\overline{CAS}$/DQM	A	(B)	C	(D)	E	(F)			
Col A, B	c1	c2	c3						
		–	–	–					
Col C, D			c1	c2	c3				
				–	–	–			
Col E, F					c1	c2	c3		
						–	–	–	
Idle							ci	ci	ci

Burst-length 2, 2 cycle latency reads, read transfers, split-read transfers

$\overline{CAS}$/DQM	A	B	C	
Col A	c1			
Col B		c1		
Col C			c1	
Idle				ci

Burst-length 1 writes, block writes, SRSs, write transfers, split-write transfers

$\overline{CAS}$/DQM	A	(B)	C	(D)	E	(F)		
Col A, B	c1	c2						
		–	–					
Col C, D			c1	c2				
				–	–			
Col E, F					c1	c2		
						–	–	
Idle							ci	ci

Burst-length 2, writes

$\overline{CAS}$/DQM		A	(B)	C	(D)	E	(F)			
Col A, B	c1	c2	c3	c4						
		–	–	–	–					
Col C, D			c1	c2	c3	c4				
				–	–	–	–			
Col E, F					c1	c2	c3	c4		
						–	–	–	–	
Idle							ci	ci	ci	ci

Burst-length 2, 3 cycle latency reads, read transfers, split-read transfers

$\overline{CAS}$/DQM	A		B		C		
Col A	c1						
		–					
Col B			c1				
				–			
Col C					c1		
						–	
Idle							ci

Burst-length 2, block writes, write transfers, split-write transfers

$\overline{CAS}$/DQM	A	(B)	C	(D)	E	(F)					
Col A, B	c1	c2	c3	c4	c5						
		–	–	–	–	–					
Col C, D			c1	c2	c3	c4	c5				
				–	–	–	–	–			
Col E, F					c1	c2	c3	c4	c5		
						–	–	–	–	–	
Idle							ci	ci	ci	ci	ci

Burst-length 2, 4 cycle latency reads, read transfers,split-read transfers.

Figure 91. SDRAM Column Pipelines

special SDRAM cycles

To initialize the SDRAM properly, the TMS320C80 performs two special SDRAM cycles after reset. The 'C80 first performs a deactivate cycle on all banks (DCAB) and then initializes the SDRAM mode register with a mode register set (MRS) cycle. The CT code input at the start of the MRS cycle determines the burst length and latency that is programmed into the SDRAM mode register (see Figure 92 and Figure 93).

special SDRAM cycles (continued)

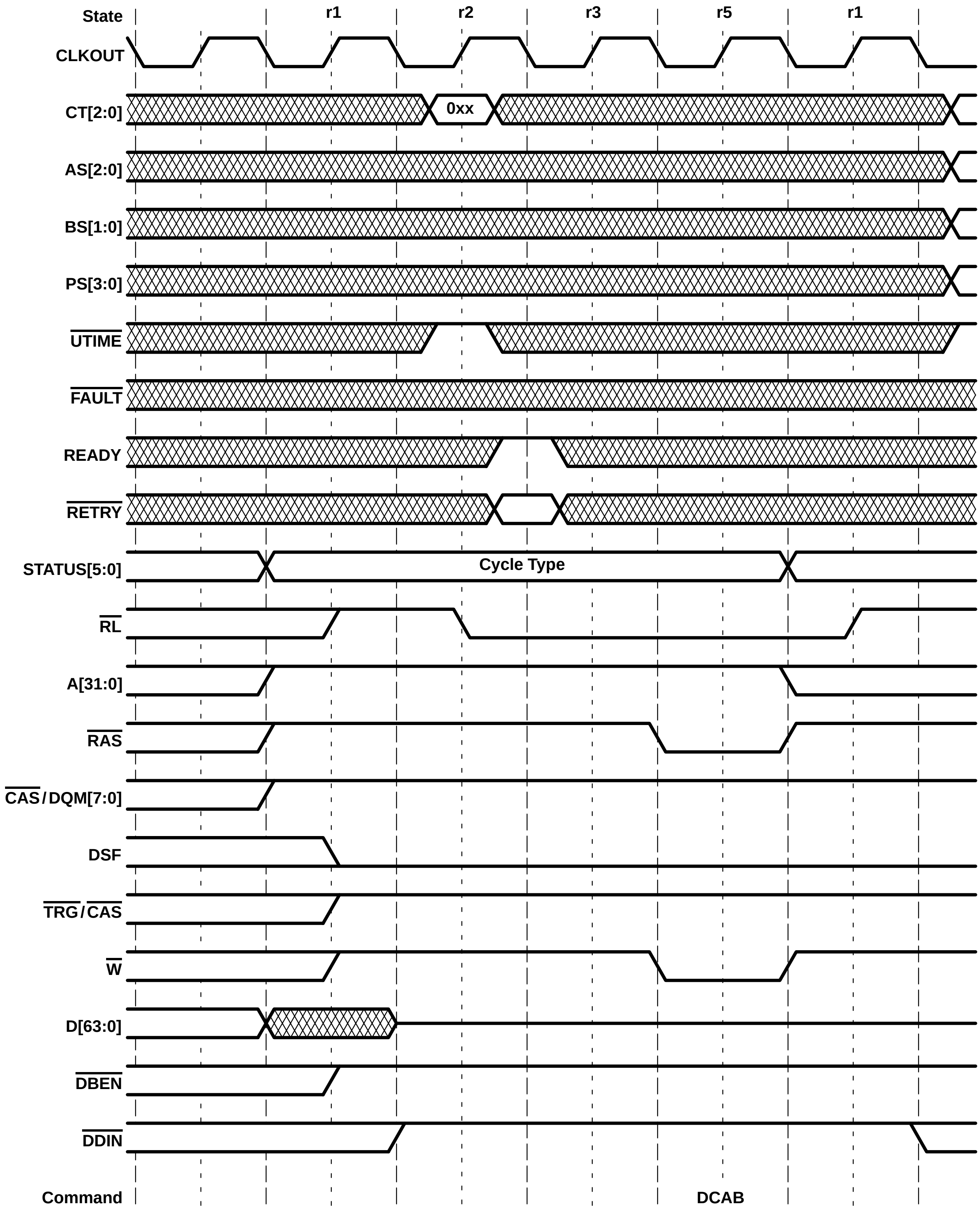


Figure 92. SDRAM Power-Up Deactivate Cycle Timing

special SDRAM cycles (continued)

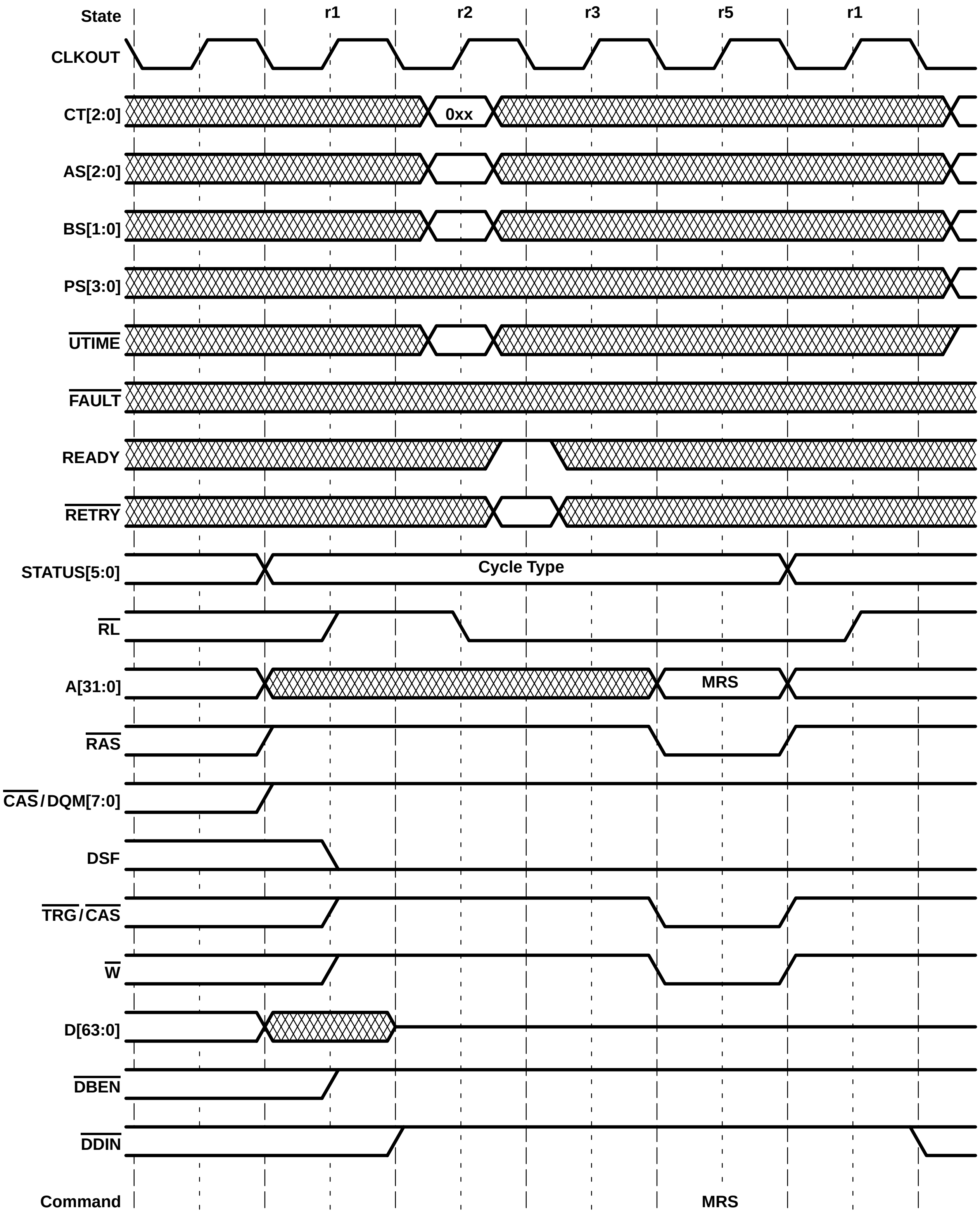


Figure 93. SDRAM Mode Register Set-Cycle Timing

SDRAM read cycles

Read cycles begin with an activate (ACTV) command to activate the bank and to select the row. The TC outputs the column address and activates the TRG/CAS strobe for each read command. For burst length 1 accesses, a read command can occur on each cycle. For burst-length 2 accesses, a read command can occur every two cycles. The TC places D[63:0] into the high-impedance state, allowing it to be driven by the memory, and latches input data during the appropriate column state. The TC always reads 64 bits and extracts and aligns the appropriate bytes. Invalid bytes for bus sizes of less than 64 bits are discarded. The CAS/DQM strobes are activated two cycles before input data is latched. If the second column in a burst is not required, then CAS/DQM is not activated. During peripheral device packet transfers, DBEN remains high (see Figure 94 through Figure 99).

SDRAM read cycles (continued)

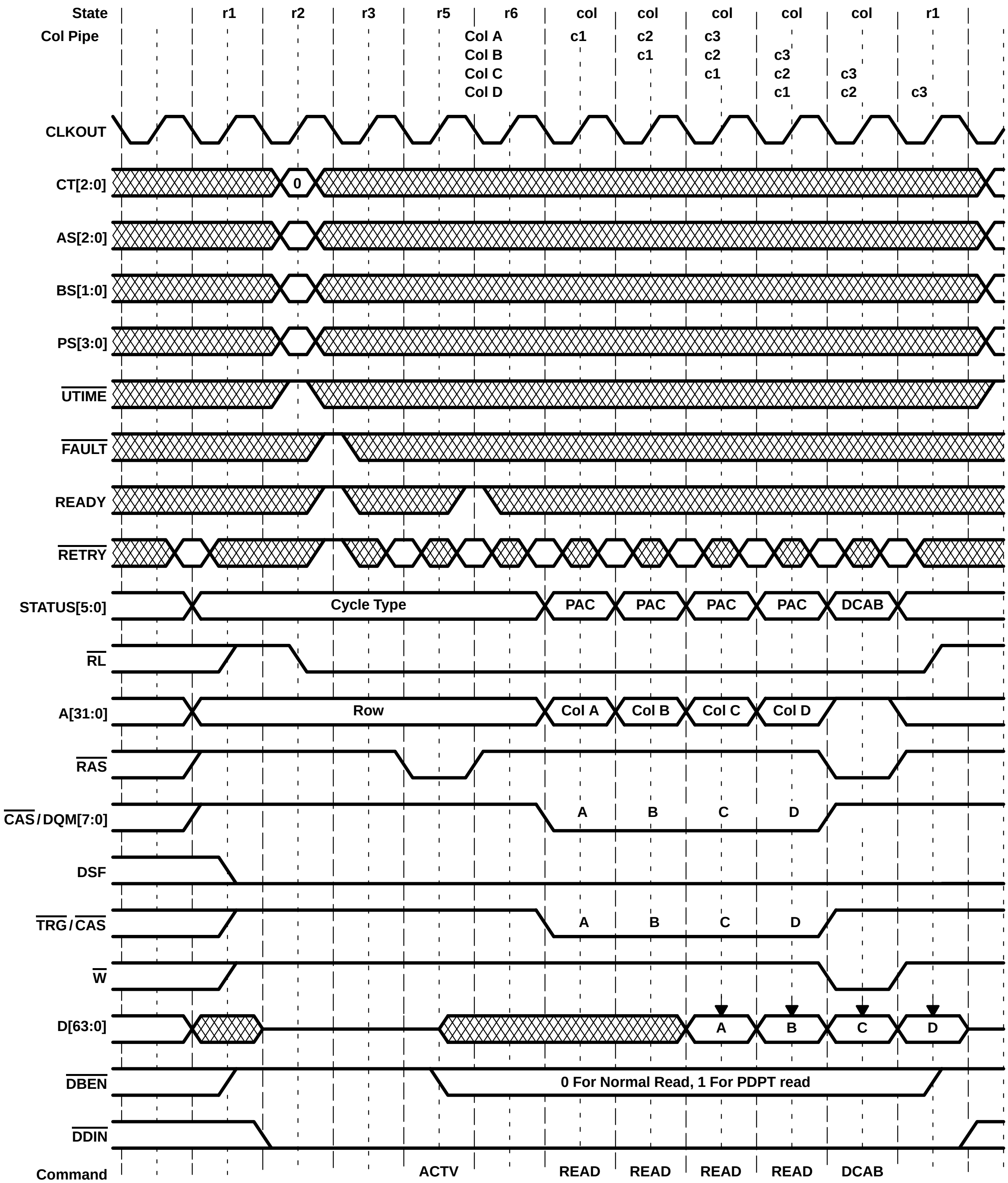


Figure 94. SDRAM Burst-Length 1, 2 Cycle Latency Read-Cycle Timing

SDRAM read cycles (continued)

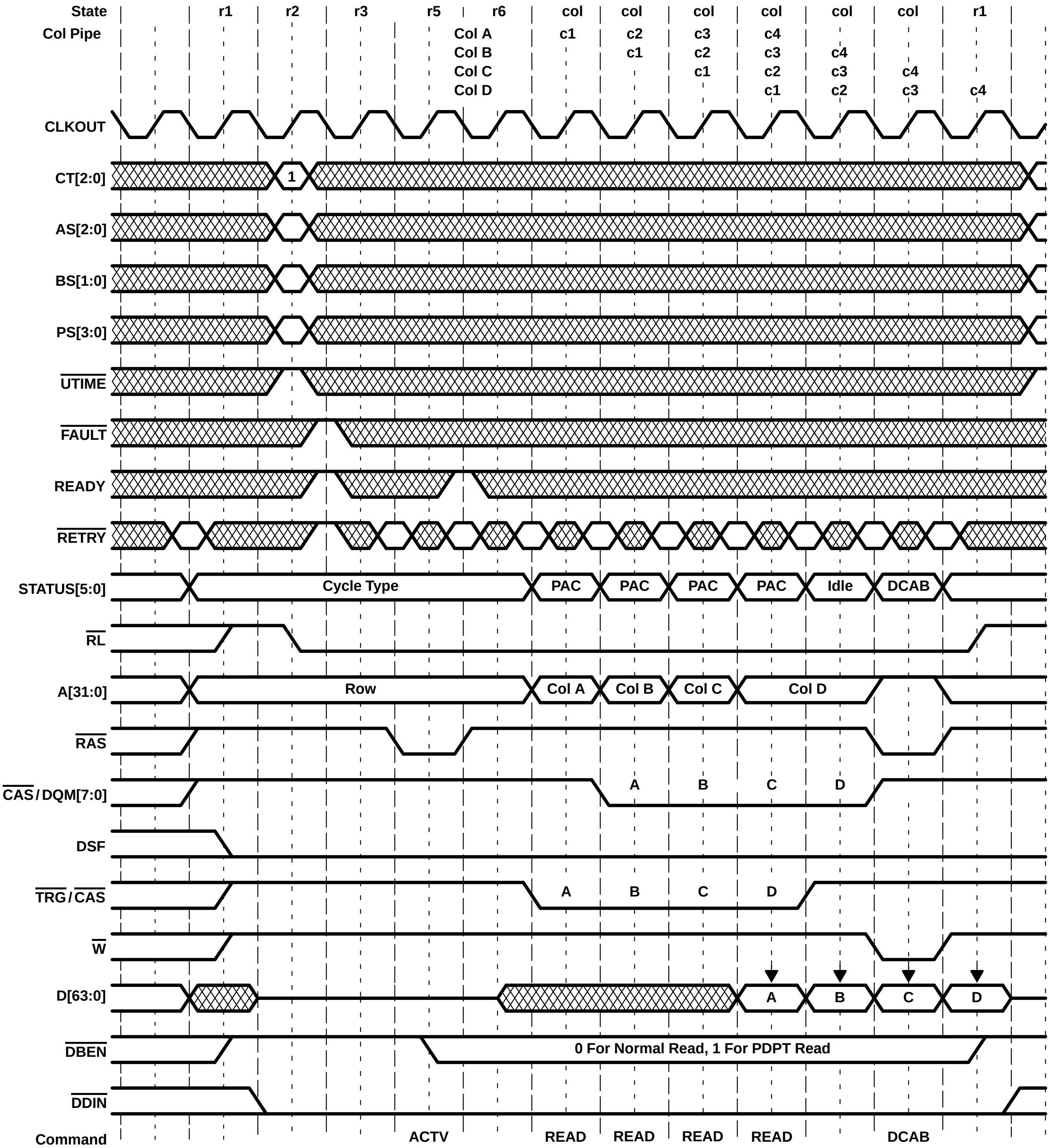


Figure 95. SDRAM Burst-Length 1, 3 Cycle Latency Read-Cycle Timing

SDRAM read cycles (continued)

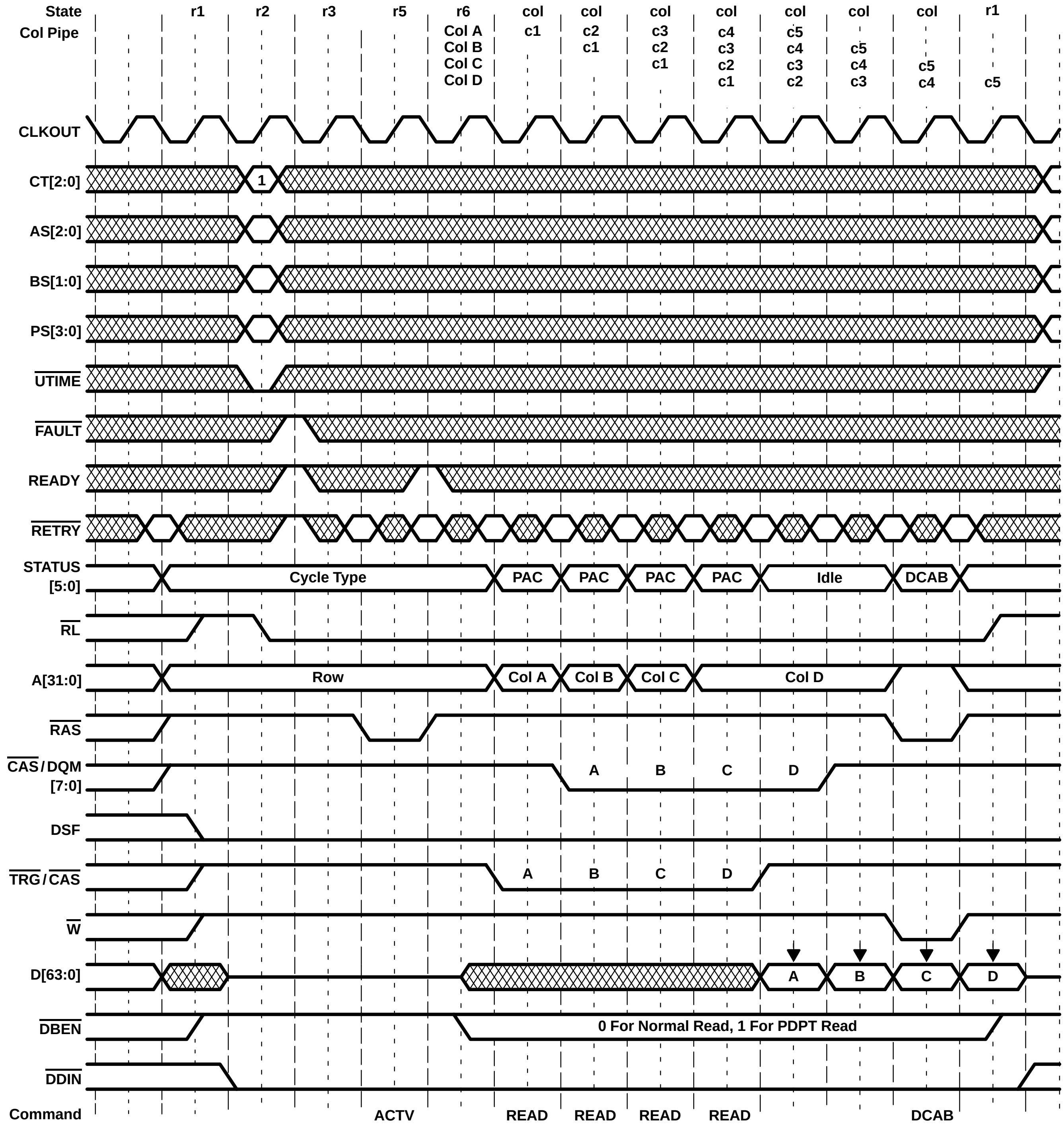


Figure 96. SDRAM Burst-Length 1, 4 Cycle Latency Read-Cycle Timing

SDRAM read cycles (continued)

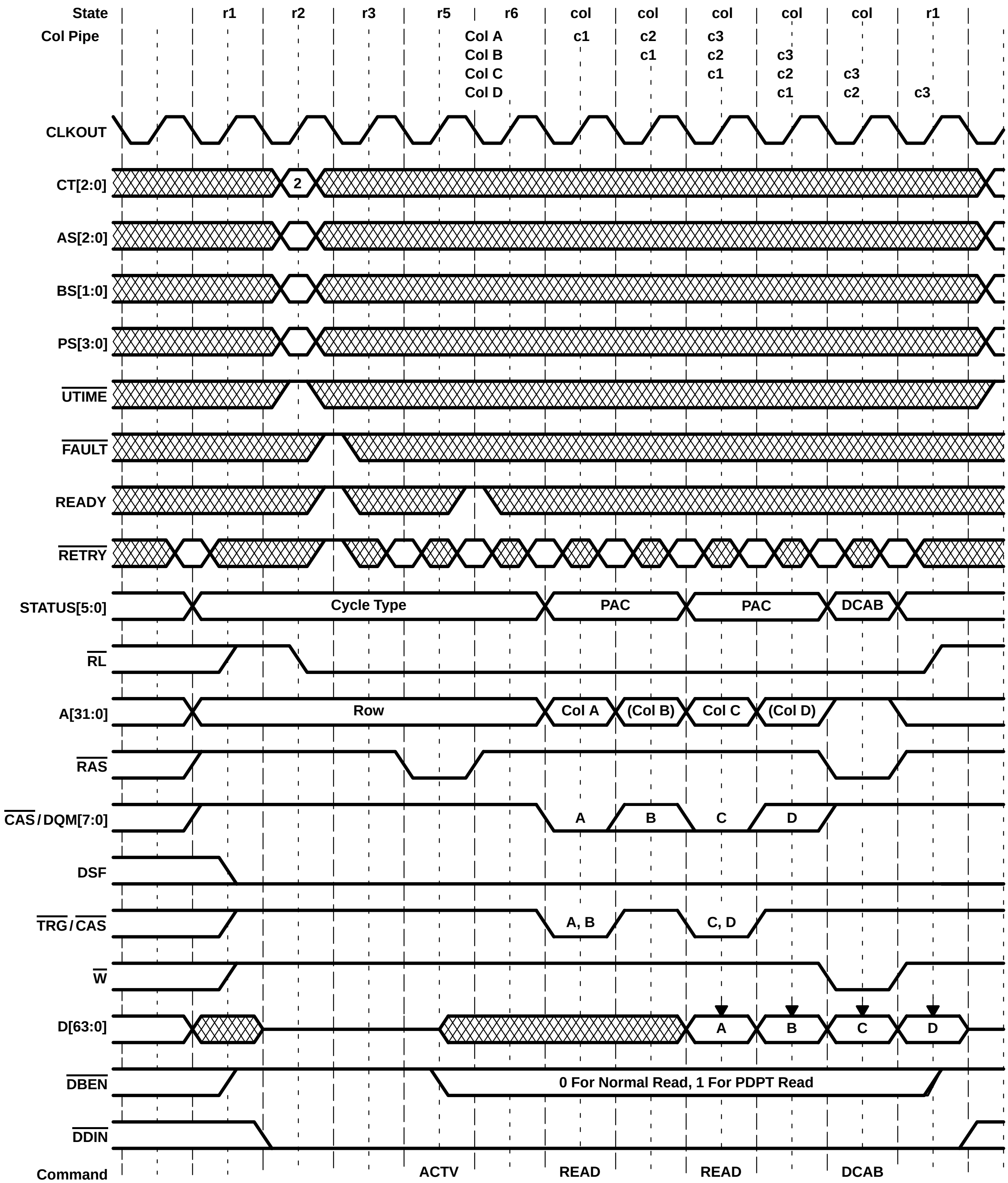


Figure 97. SDRAM Burst-Length 2, 2 Cycle Latency Read-Cycle Timing

SDRAM read cycles (continued)

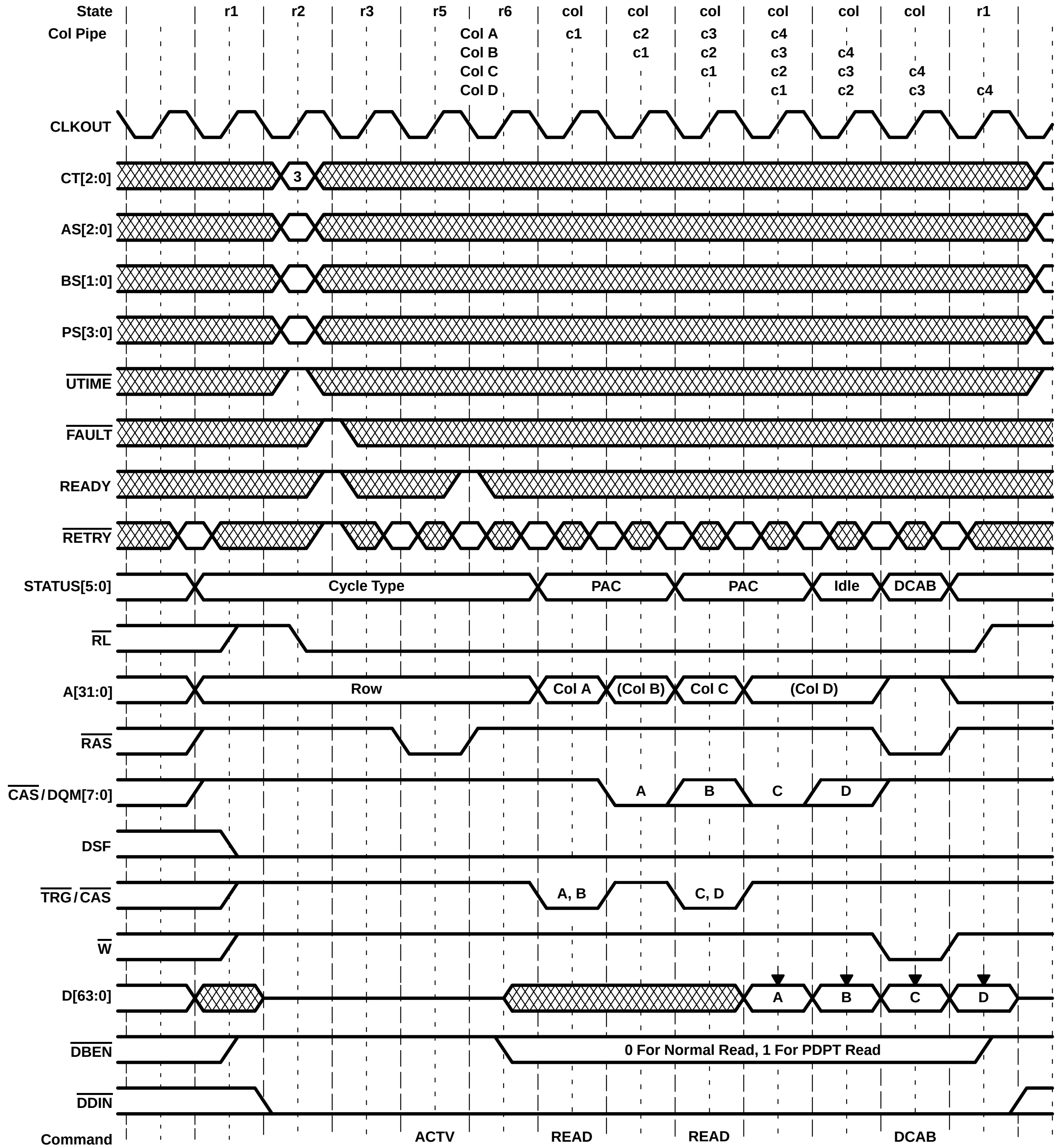


Figure 98. SDRAM Burst-Length 2, 3 Cycle Latency Read-Cycle Timing

SDRAM read cycles (continued)

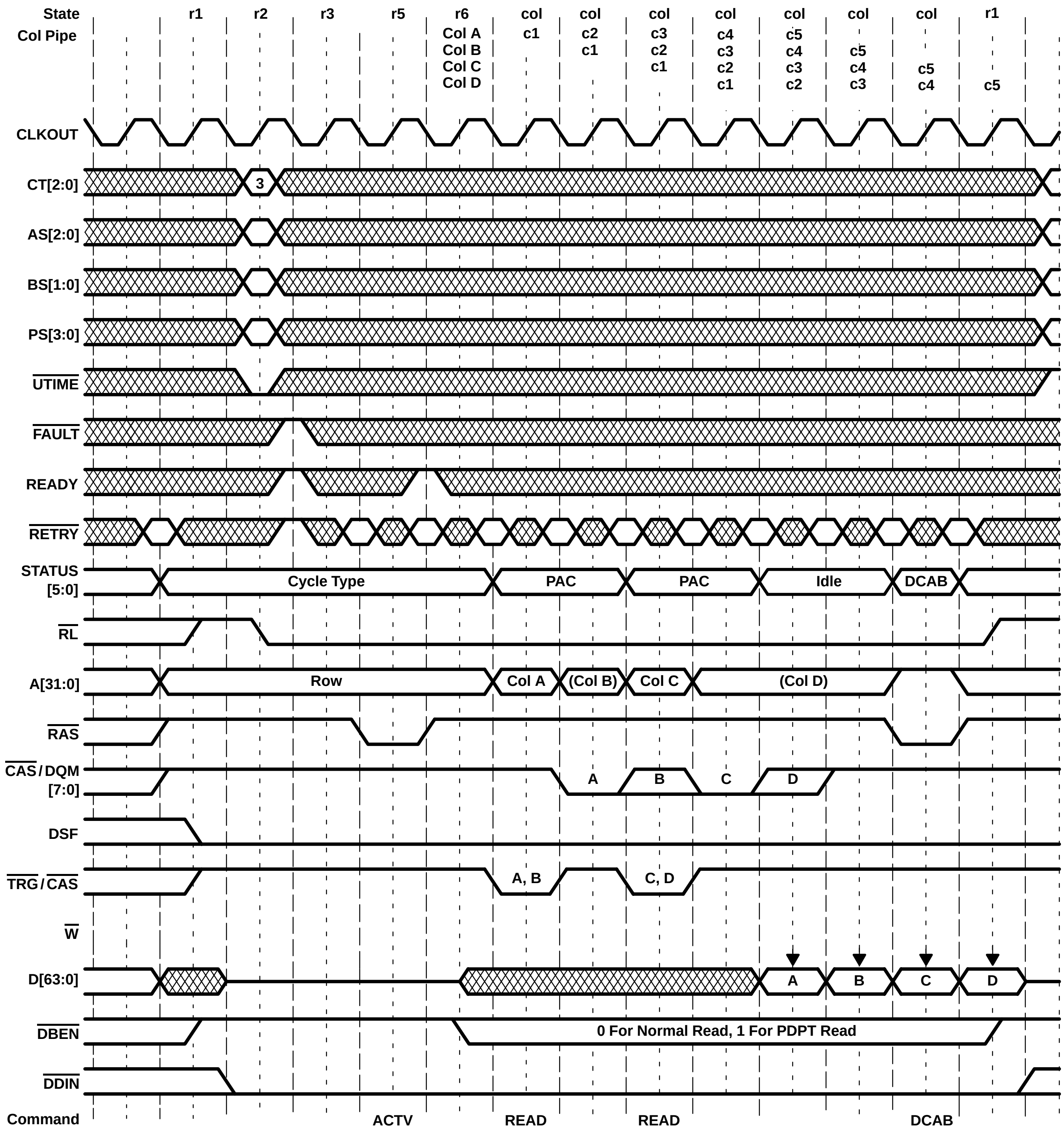


Figure 99. SDRAM Burst-Length 2, 4 Cycle Latency Read-Cycle Timing

SDRAM write cycles

Write cycles begin with an activate (ACTV) command to activate the bank and select the row. The TC outputs the column address and activates the $\overline{TRG}/\overline{CAS}$ and $\overline{W}$ strobes for each write command. For burst-length 1 accesses, a write command can occur on each cycle. For burst-length 2 accesses, a write command can occur every two cycles. The TC drives data out on D[63:0] during each cycle of an active-write command and indicates valid bytes by driving the appropriate $\overline{CAS}$/DQM strobes low. During peripheral device packet transfers, $\overline{DBEN}$ remains high and D[63:0] are placed in the high-impedance state so that the peripheral can drive data into the memories. For SDRAM write cycles, see Figure 100 and Figure 101.

SDRAM write cycles (continued)

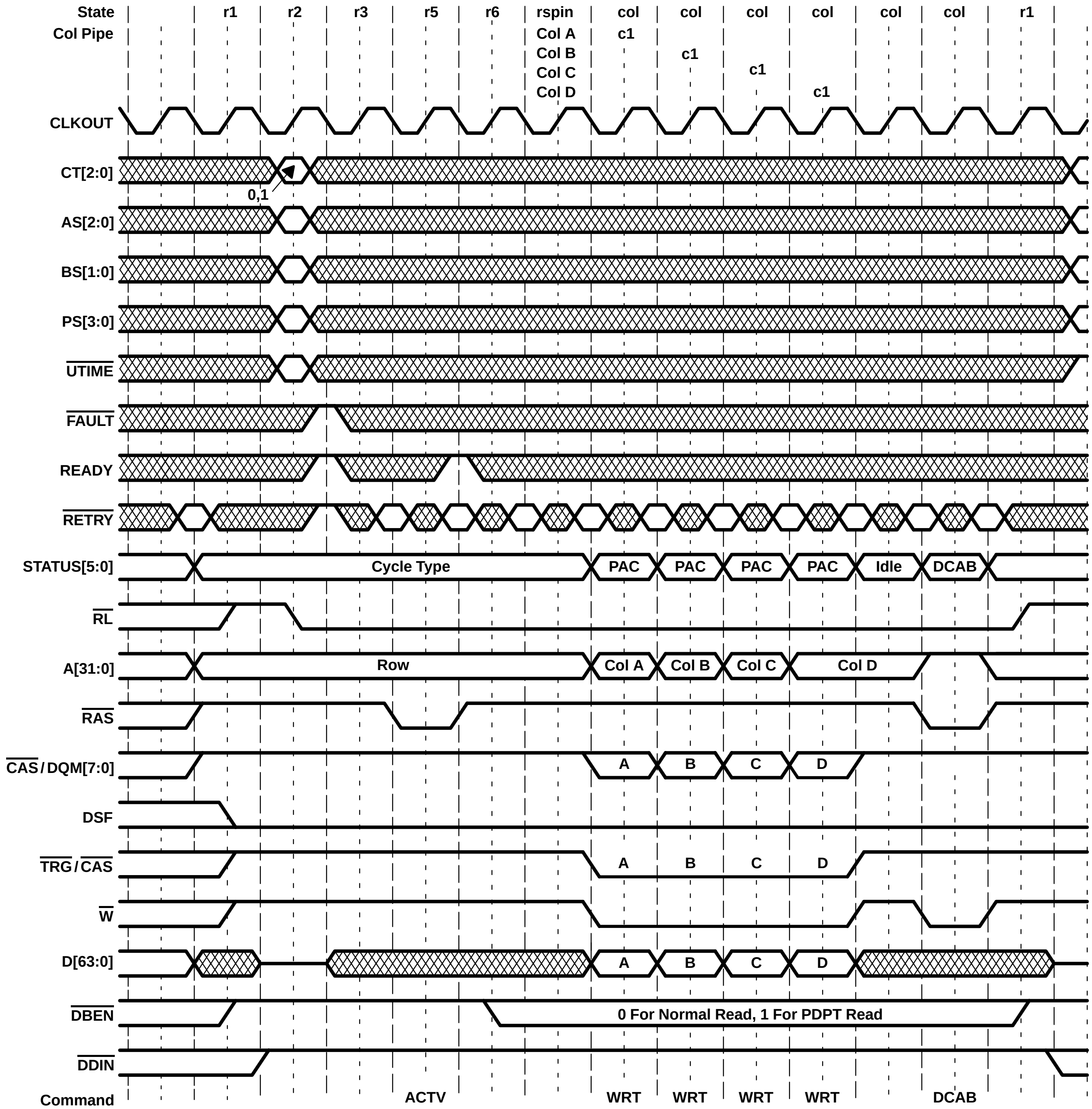


Figure 100. SDRAM Burst-Length 1 Write-Cycle Timing

SDRAM write cycles (continued)

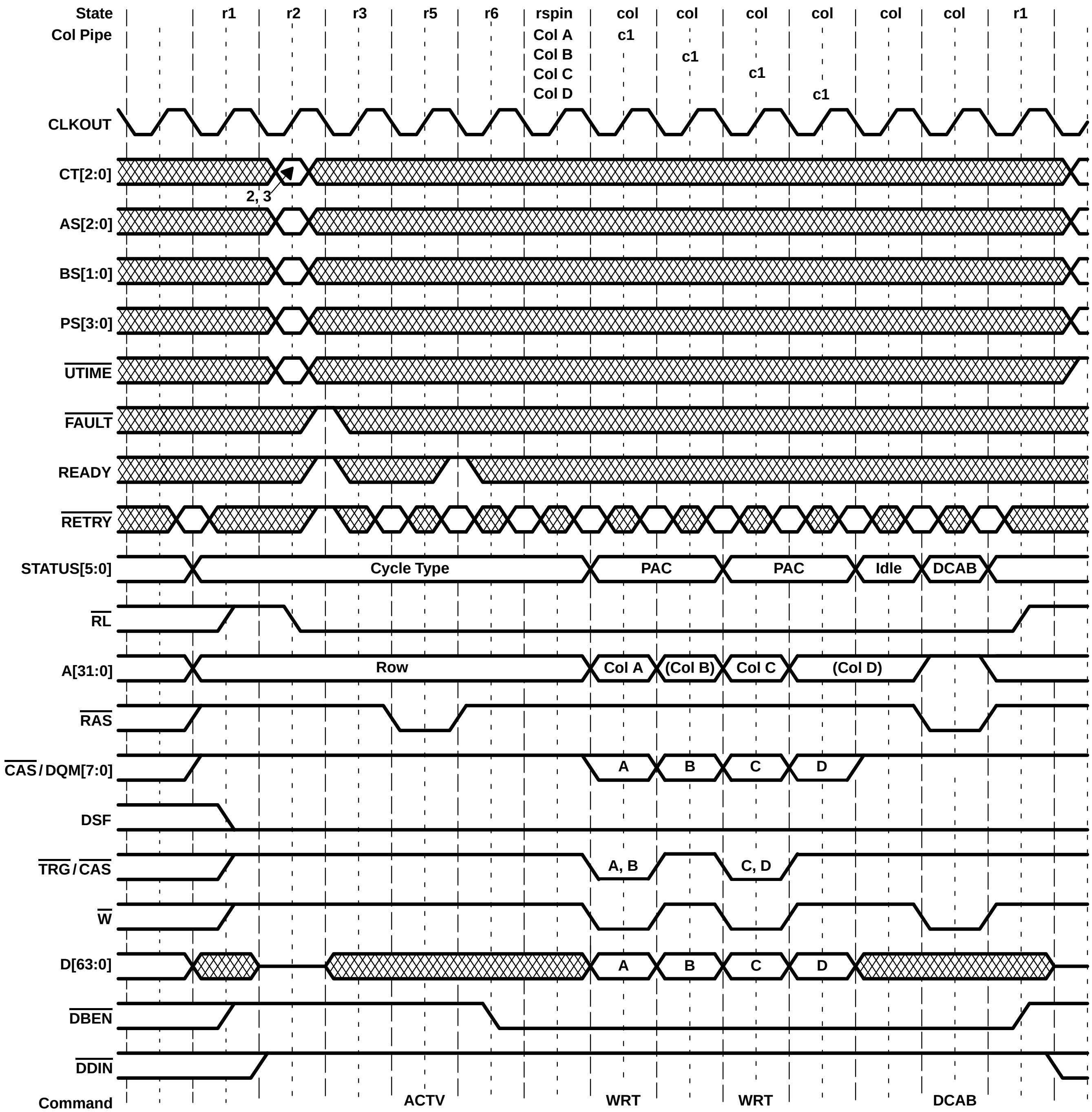


Figure 101. SDRAM Burst-Length 2 Write-Cycle Timing

special register set cycles

Special register set (SRS) cycles are used to program control registers within an SVRAM or SGRAM. The ’C80 only supports programming of the color register for use with block writes. The cycle is similar to a single burst length 1 write cycle but DSF is driven high. The values output on the ’C80 address bits cause the color register to be selected as shown in Figure 102 (see Figure 103).

special register set cycles (continued)

SDRAM Address Pin	BS	A8	A7	A6	A5	A4	A3	A2	A1	A0
SDRAM Function	0	0	0	LC	LM	LS	Stop Register			
TMS320C80 Output Value	0	0	0	1	0	0	0	0	0	0

Figure 102. Special-Register-Set Value

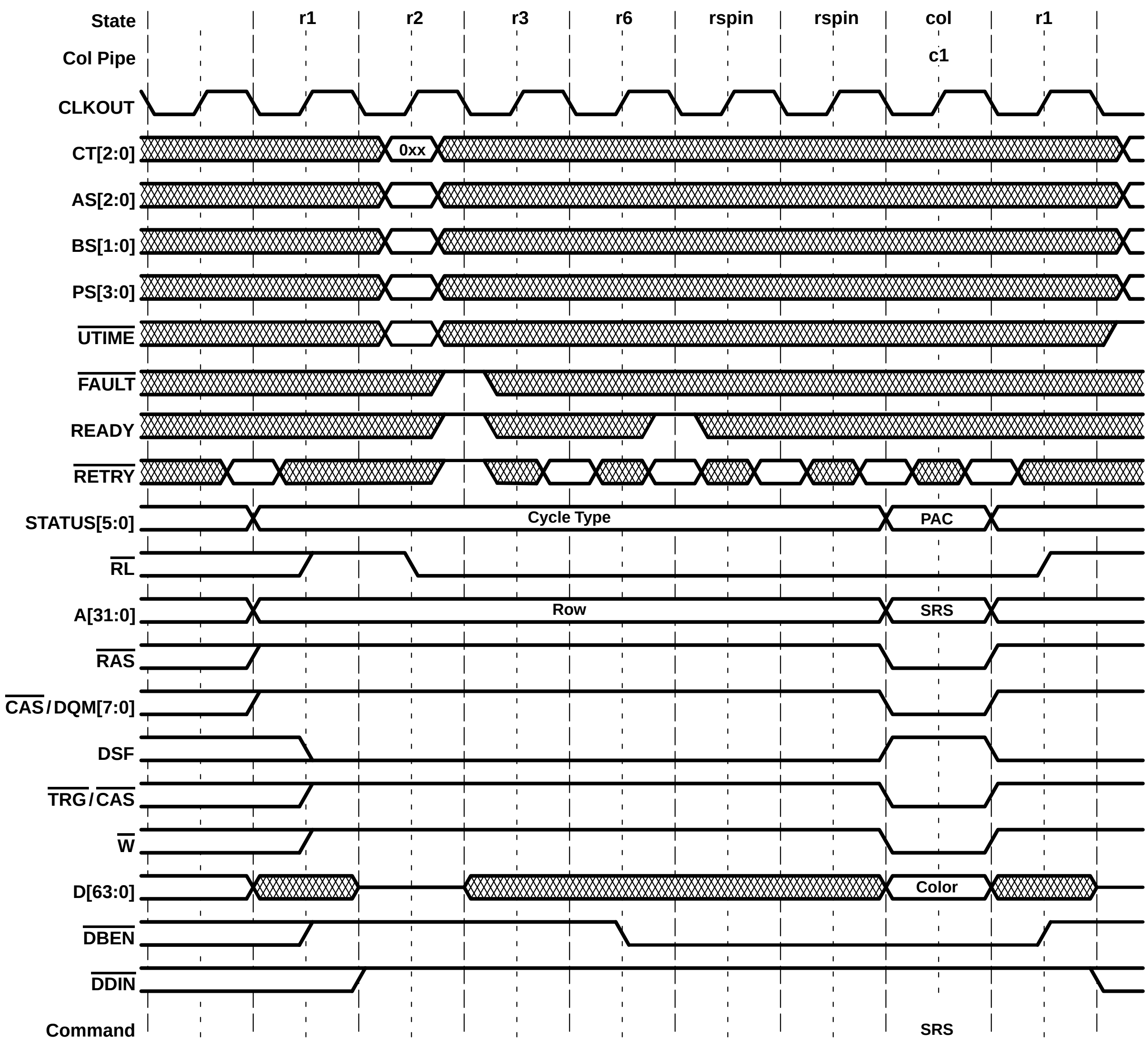


Figure 103. SDRAM SRS-Cycle Timing

SDRAM block-write cycles

Block-write cycles allow SVRAMs and SGRAMs to write a stored color value to multiple column locations in a single access. Block-write cycles are similar to write cycles except that DSF is driven high to indicate a block-write command. Because burst is not supported for block write, burst length 2 accesses generate a single block-write every other clock cycle (see Figure 104 and Figure 105).

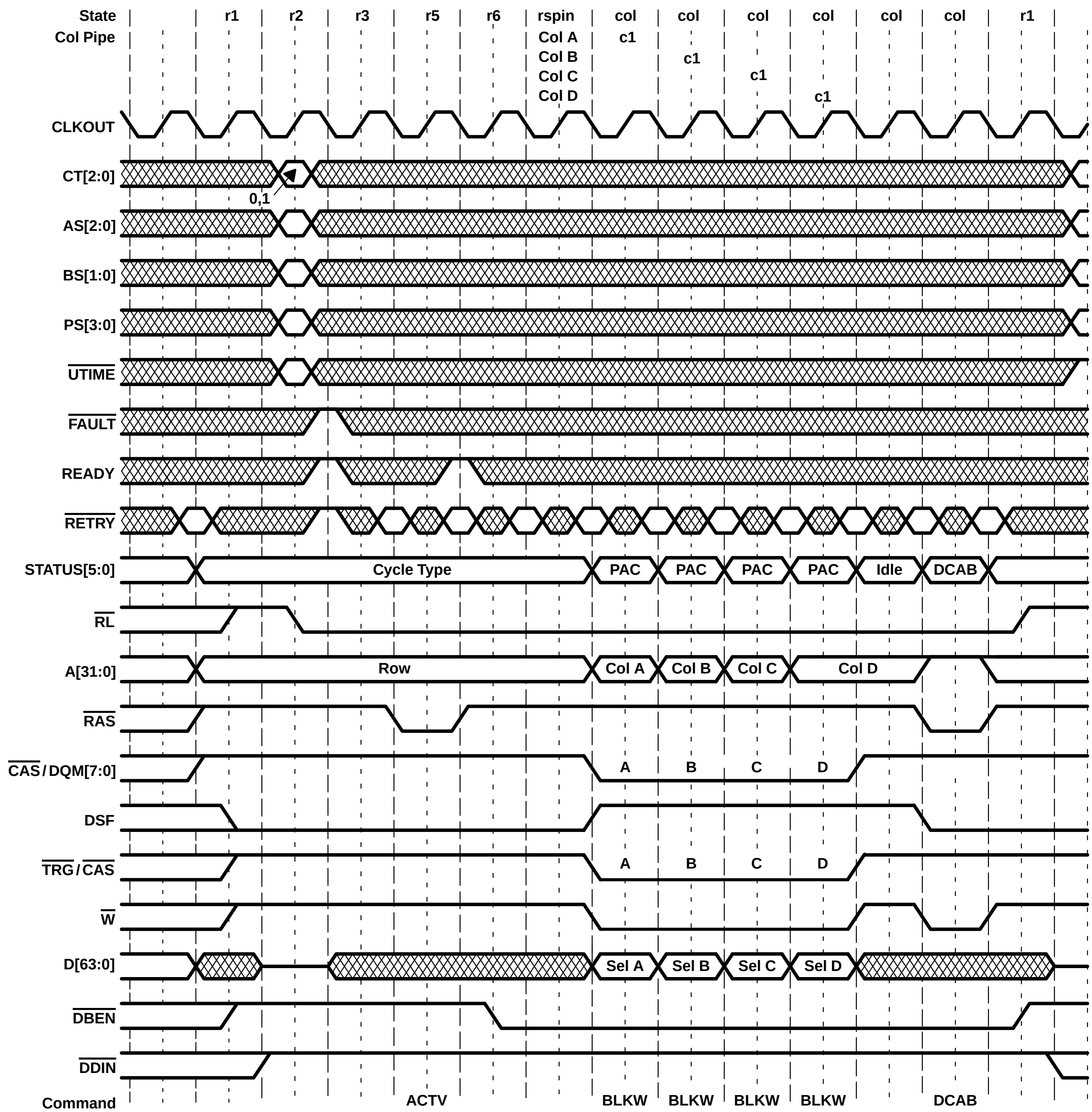


Figure 104. SDRAM Burst-Length 1 Block-Write Cycle Timing

SDRAM block-write cycles (continued)

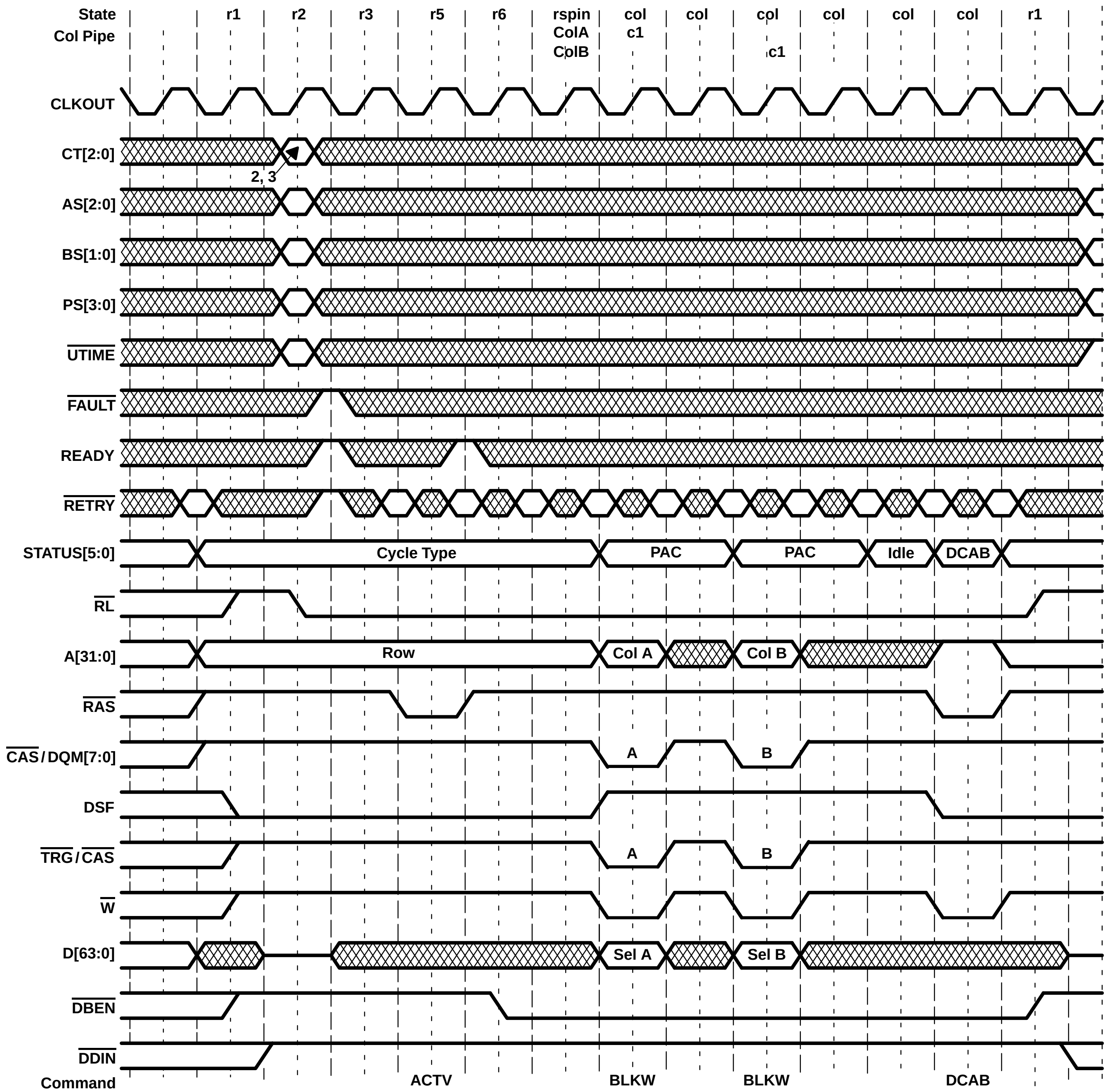


Figure 105. SDRAM Burst-Length 2 Block-Write Cycle Timing

SVRAM transfer cycles

The SVRAM read- and write-transfer cycles transfer data between the SVRAM memory-array and the serial register (SAM). The TMS320C80 supports both normal and split transfers for SVRAMs. Read-and split-read transfers resemble a standard read cycle. Write-and split-write transfers resemble a standard write cycle. Because the 'C80's $\overline{\text{TRG}}$ output is used as $\overline{\text{CAS}}$, external logic must generate a $\overline{\text{TRG}}$ signal (by decoding STATUS) to enable the SVRAM transfer cycle. The value output on A[31:0] at column time represents the SAM tap point (see Figure 106 through Figure 113).

SVRAM transfer cycles (continued)

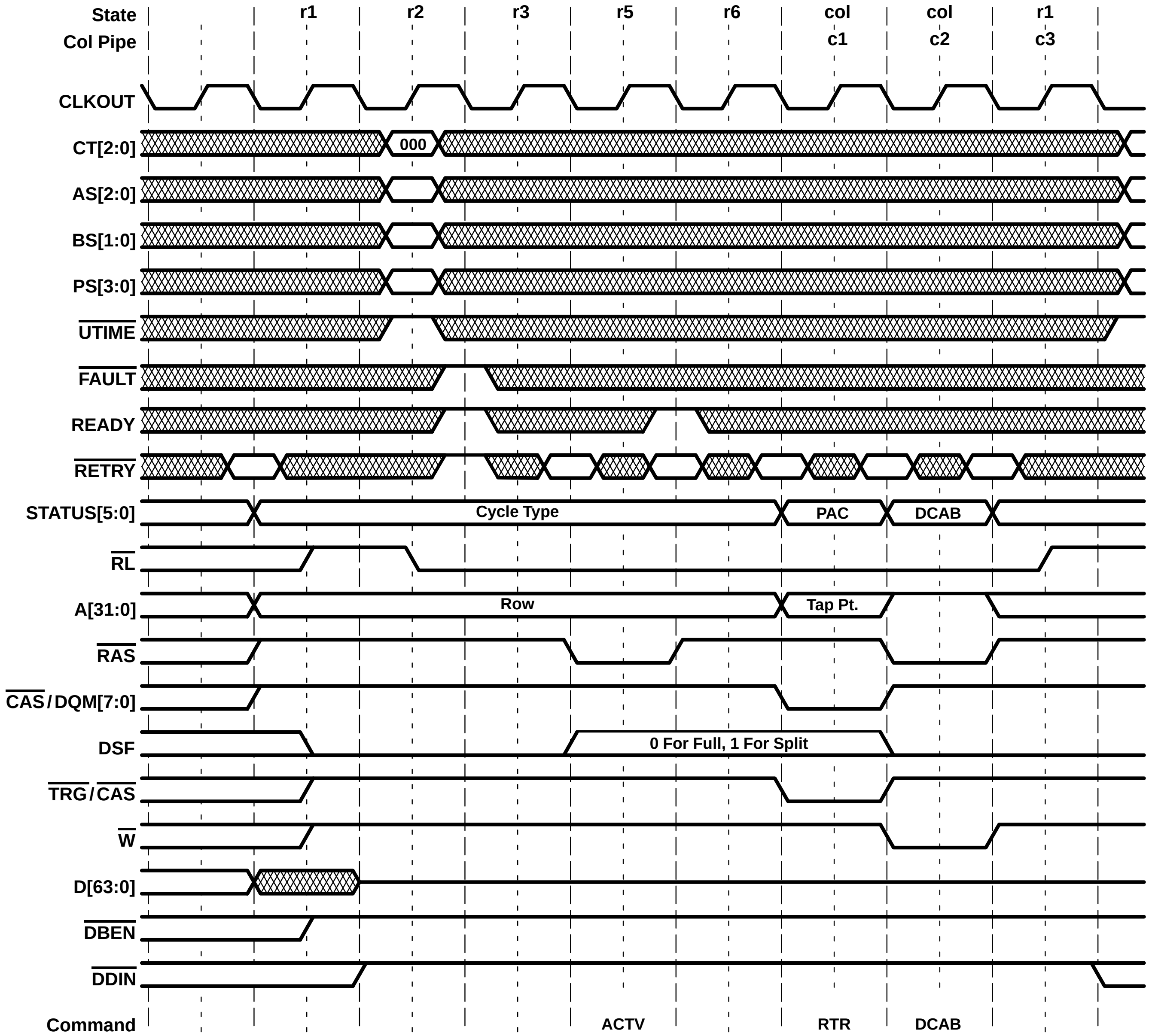


Figure 106. SVRAM Burst-Length 1, 2 Cycle Latency Read-Transfer Cycle Timing

SVRAM transfer cycles (continued)

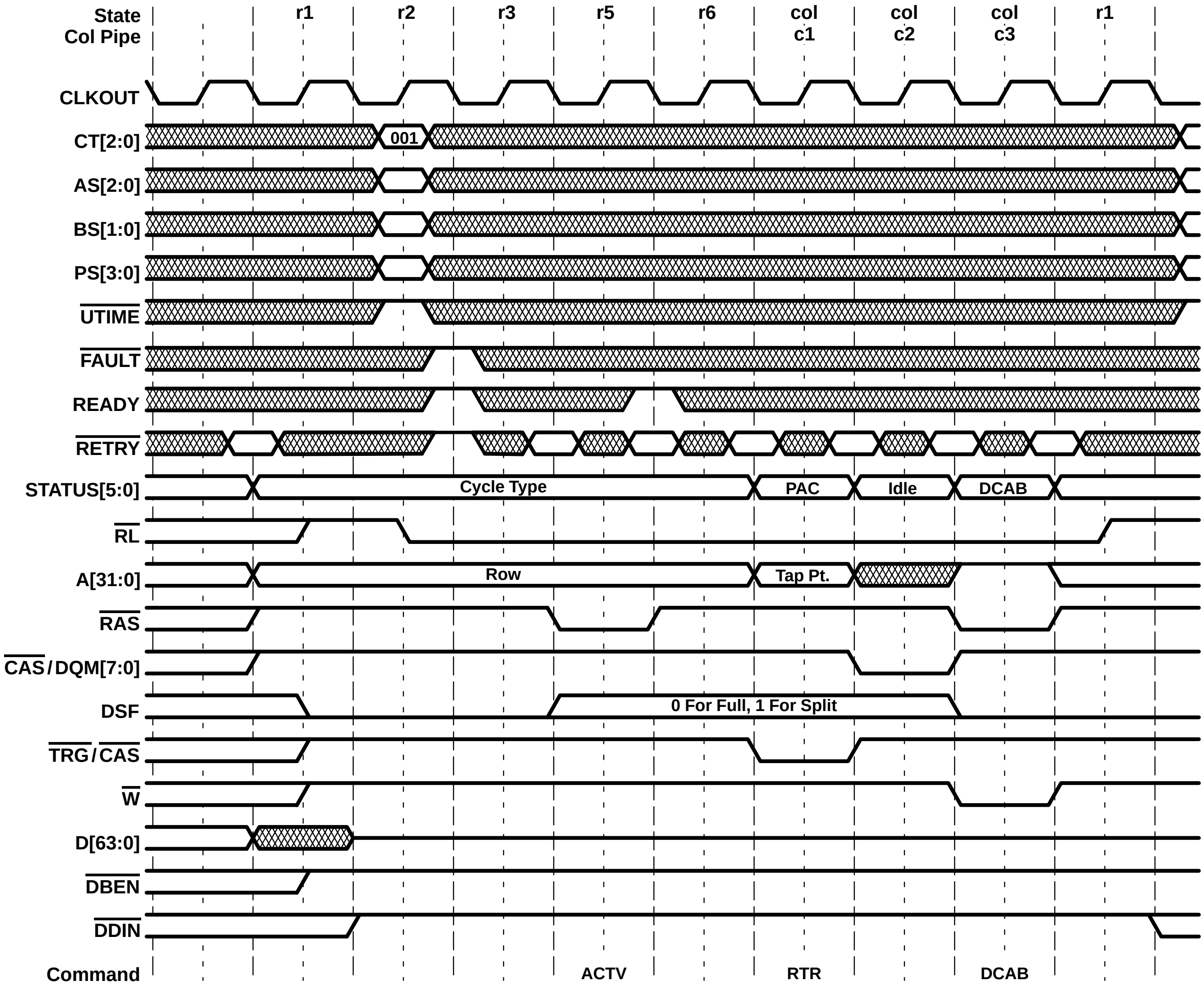


Figure 107. SVRAM Burst-Length 1, 3 Cycle Latency Read-Transfer Cycle Timing

SVRAM transfer cycles (continued)

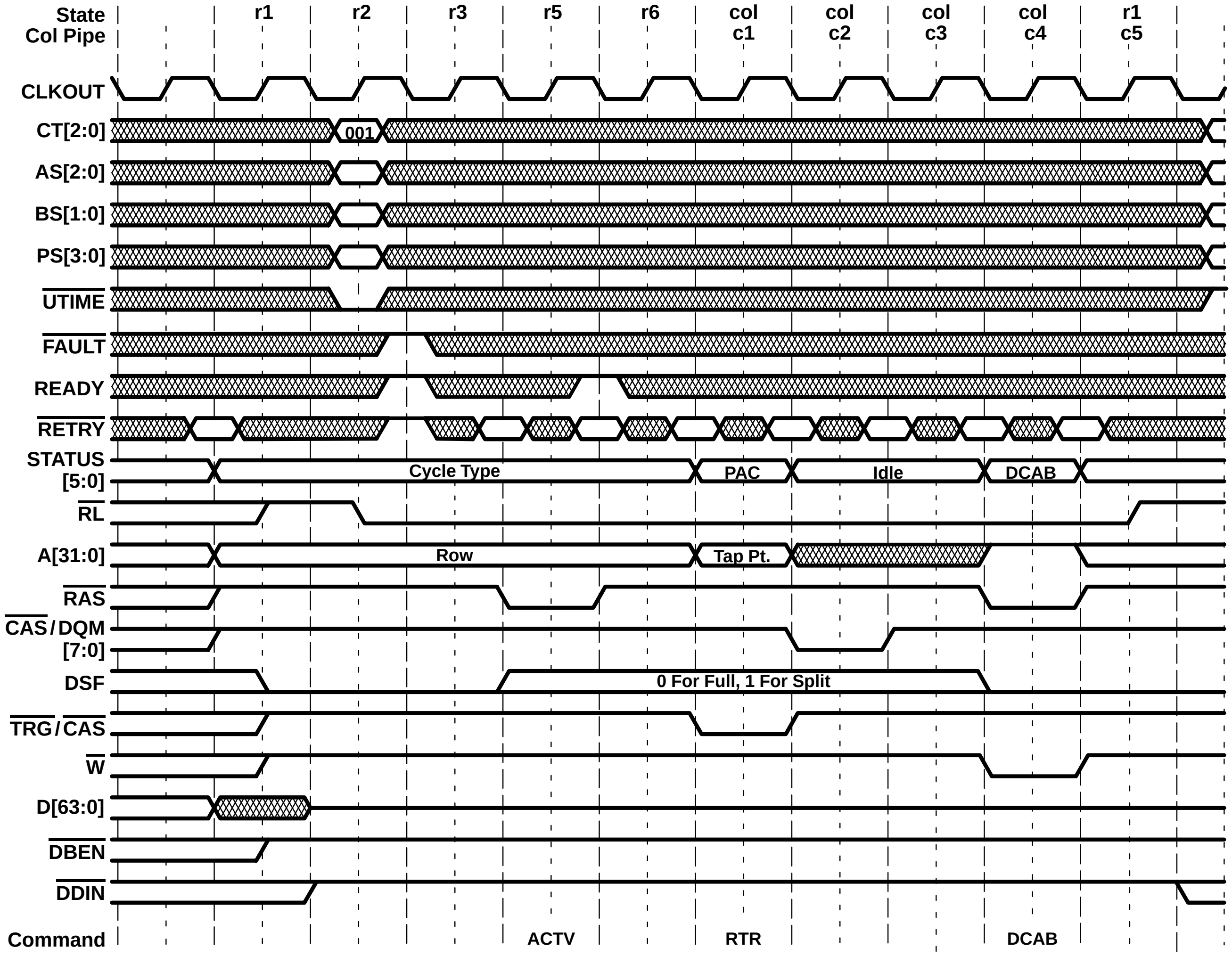


Figure 108. SVRAM Burst-Length 1, 4 Cycle Latency Read-Transfer Cycle Timing

SVRAM transfer cycles (continued)

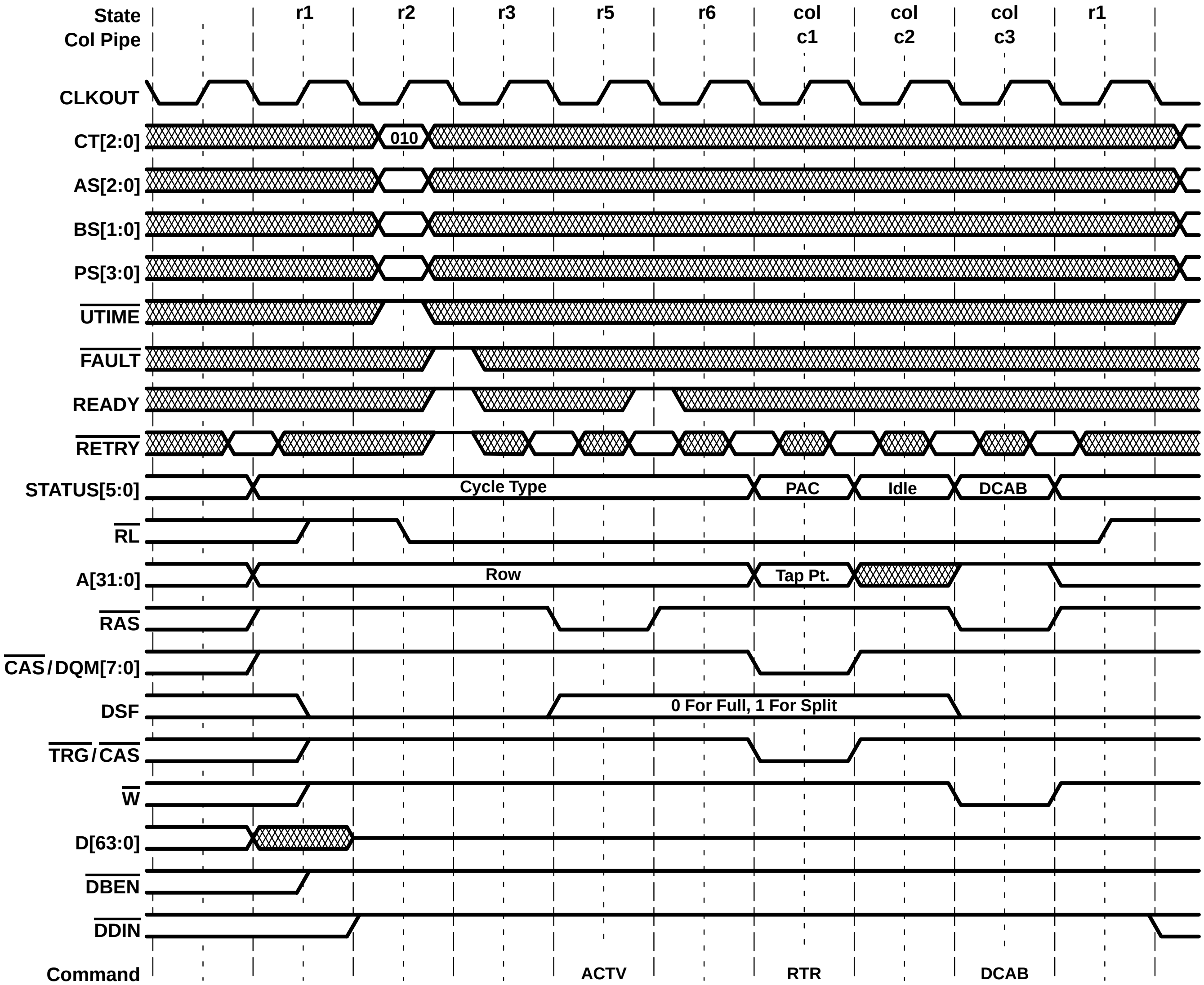


Figure 109. SVRAM Burst-Length 2, 2 Cycle Latency Read-Transfer Cycle Timing

SVRAM transfer cycles (continued)

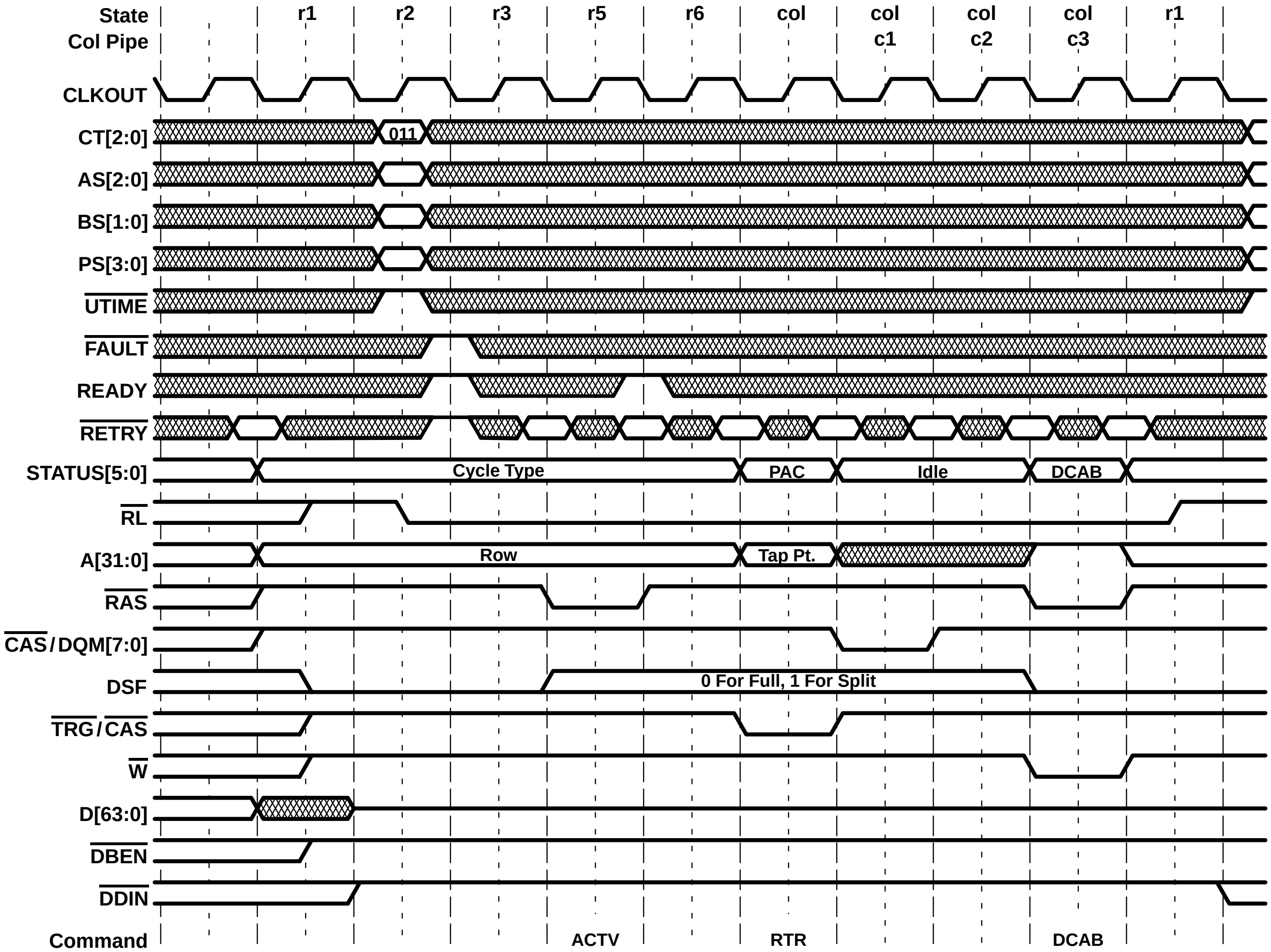


Figure 110. SVRAM Burst-Length 2, 3 Cycle Latency Read-Transfer Cycle Timing

SVRAM transfer cycles (continued)

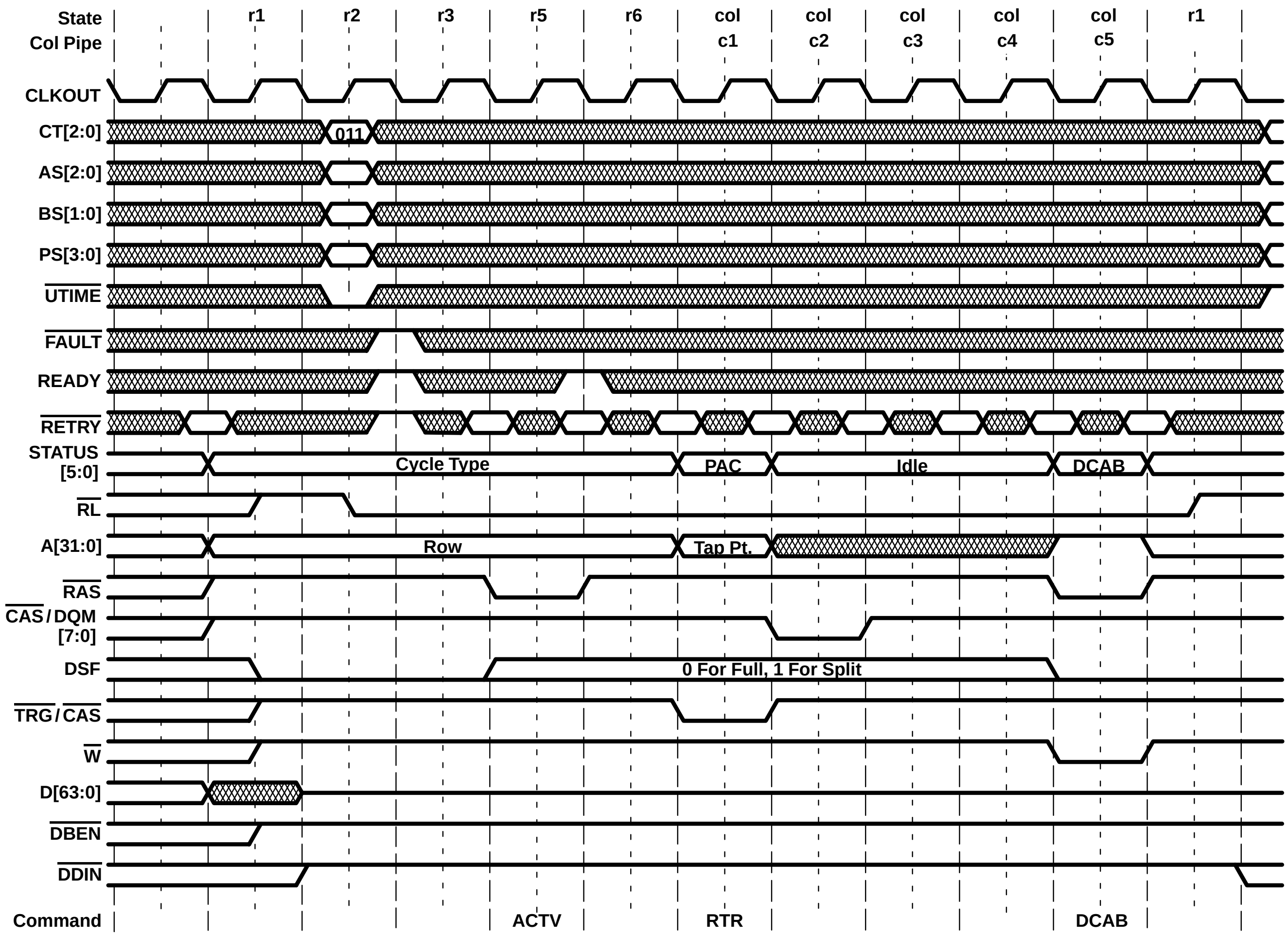


Figure 111. SVRAM Burst-Length 2, 4 Cycle Latency Read-Transfer Cycle Timing

SVRAM transfer cycles (continued)

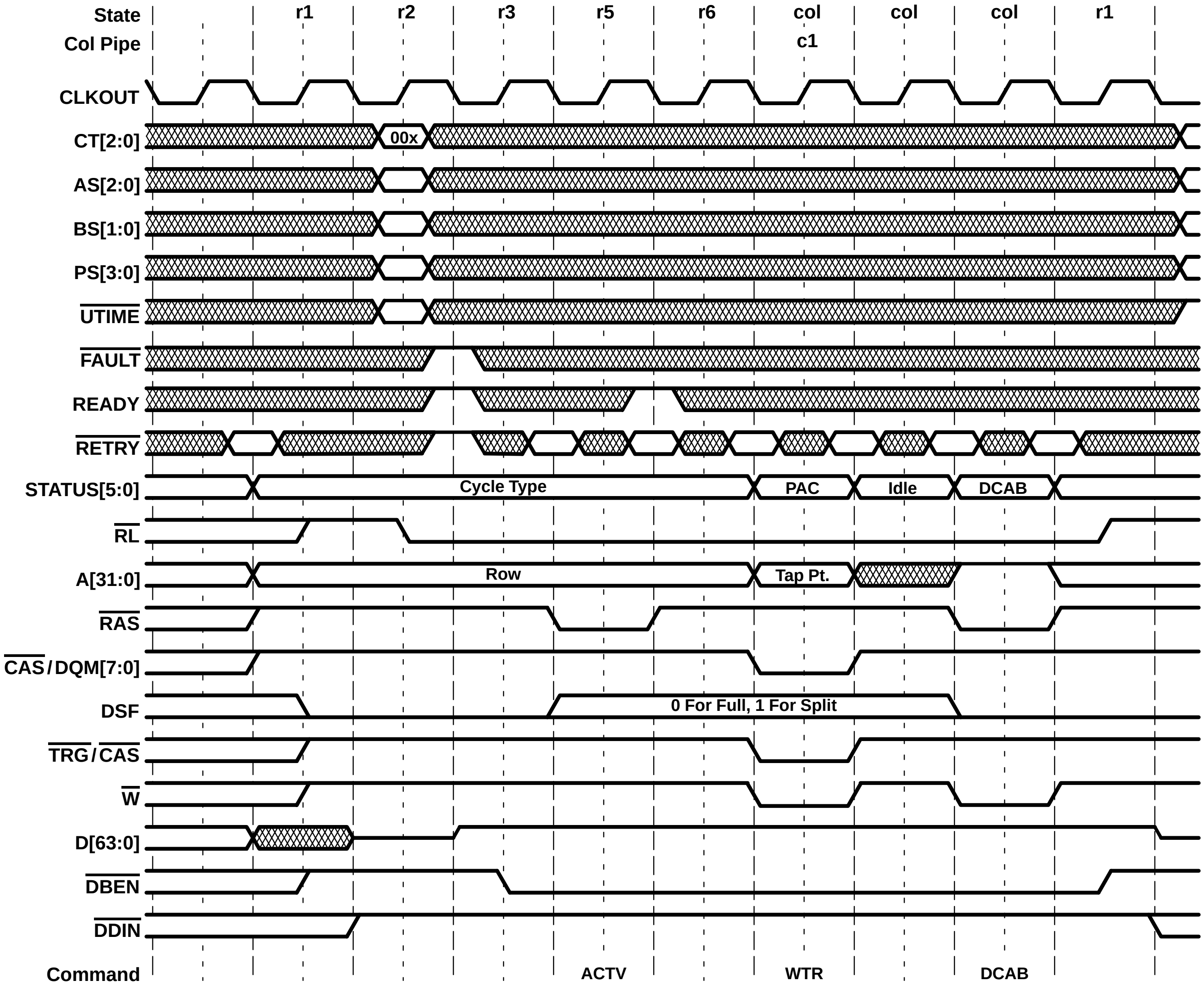


Figure 112. SVRAM Burst-Length 1, Write-Transfer Cycle Timing

SVRAM transfer cycles (continued)

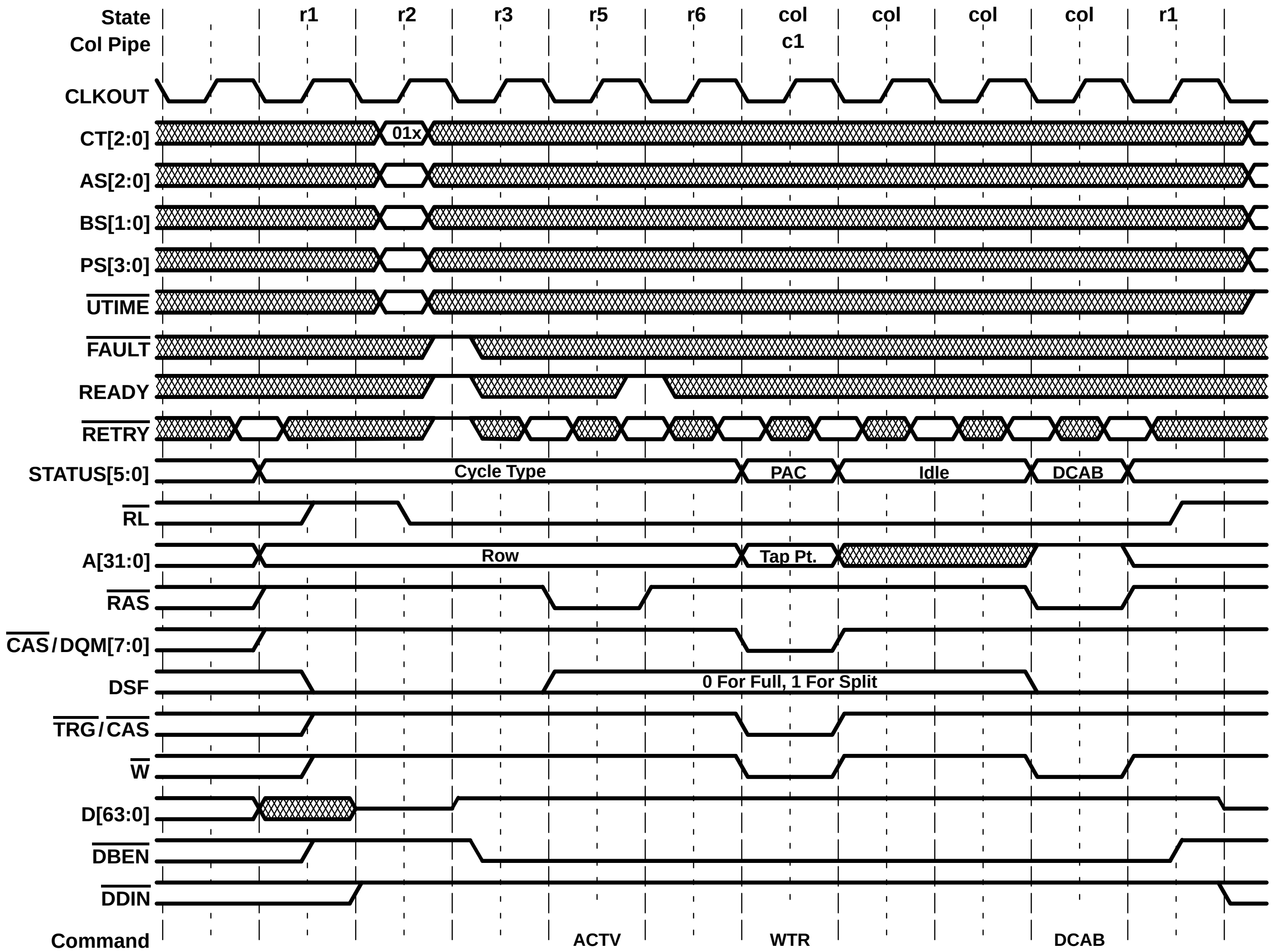


Figure 113. SVRAM Burst-Length 2, Write-Transfer Cycle Timing

SDRAM refresh cycle

The SDRAM refresh cycle is performed when the TC receives an SDRAM-cycle timing input (CT=0xx) at the start of a refresh cycle. The RAS and TRG/CAS outputs are driven low for one cycle to strobe a refresh command (REFR) into the SDRAM. The refresh address is generated internal to the SDRAM. The ’C80 outputs a 16-bit pseudo-address (used for refresh bank decode) on A[31:16] and drives A[15:0] low (see Figure 114).

SDRAM refresh cycle (continued)

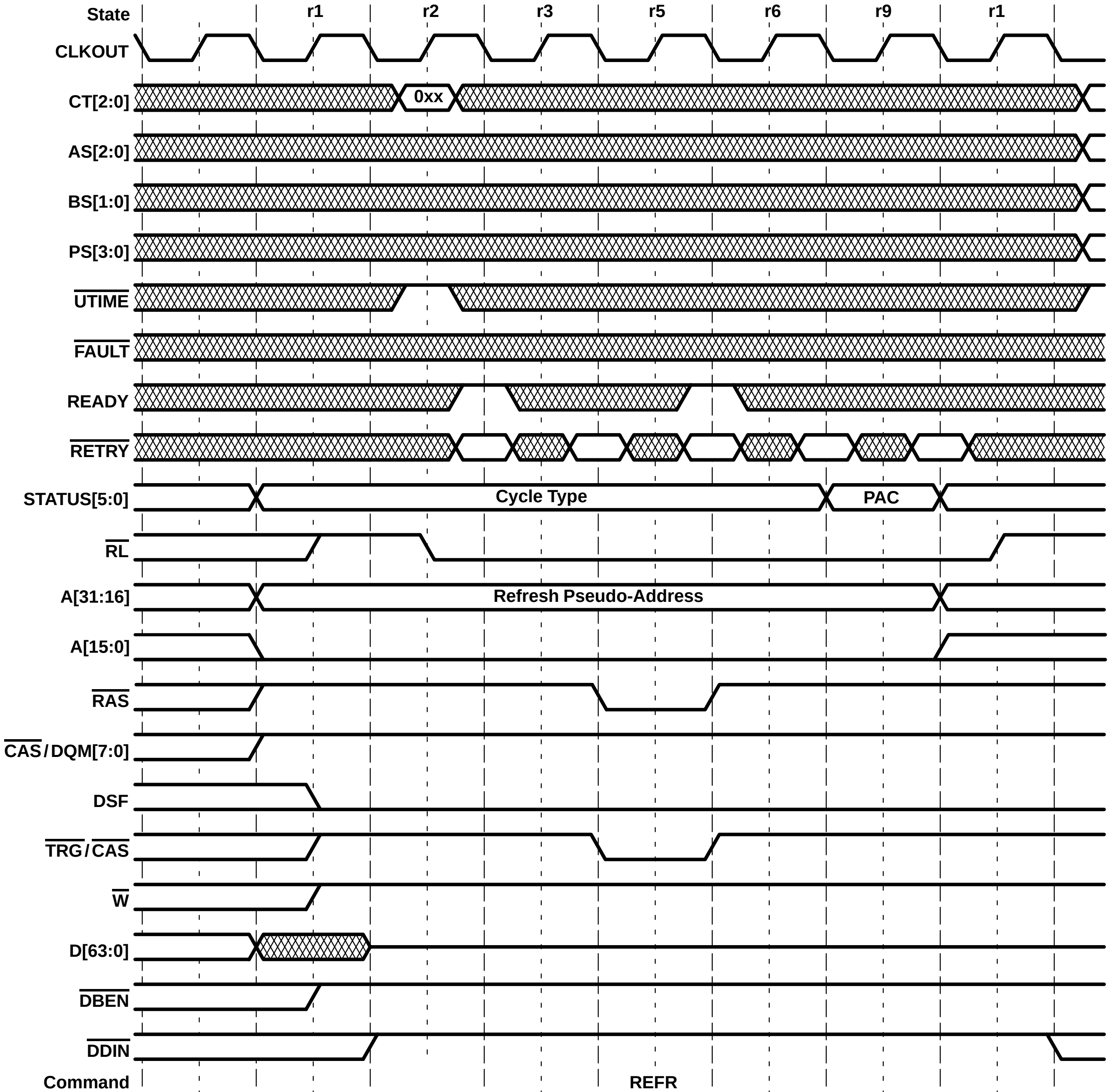


Figure 114. SDRAM Refresh-Cycle Timing

host interface

The ’C80 contains a simple four-pin mechanism by which a host or another device can gain control of the ’C80 local memory bus. The HREQ input can be driven low by the host to request the ’C80’s bus. Once the TC has completed the current memory access, it places the local bus (except CLKOUT) into a high-impedance state. It then drives the HACK output low to indicate that the host device owns the bus and can drive it. The REQ[1:0] outputs reflect the highest priority cycle request being received internally by the TC. The host can monitor these outputs to determine if it needs to relinquish the local bus back to the ’C80 (see Table 37).

Table 37. TC Priority Cycles

REQ[1:0]	ASSOCIATED INTERNAL TC REQUEST
11	SRT, urgent refresh, XPT, or VCPT
10	Cache / DEA request, urgent packet transfer
01	High-priority packet transfer
00	Low-priority packet transfer, trickle refresh, idle

device reset

The TMS320C80 is reset when the RESET input is driven low. The ’C80 outputs immediately go into a high-impedance state with the exception of CLKOUT, HACK, and REQ[1:0]. While RESET is low, all internal registers are set to their default values and internal logic is reset.

On the rising edge of RESET, the state of UTIME is sampled to determine if big-endian (UTIME = 0) or little-endian (UTIME = 1) operation is selected. Also on the rising edge of RESET, the state of HREQ is sampled to determine if the master processor comes up running (HREQ = 0) or halted (HREQ = 1).

Once RESET is high, the ’C80 drives the high-impedance signals to their inactive values. The TC then performs 32 refresh cycles to initialize system memory. If, during initialization refresh, the TC receives an SDRAM cycle timing code (CT = 0xx), it performs an SDRAM DCAB cycle and a MRS cycle to initialize the SDRAM, and then continues the refresh cycles.

After completing initialization refresh, if the MP is running, the TC performs its instruction-cache-fill request to fetch the cache block beginning at 0xFFFFFFC0. This block contains the starting MP instruction located at 0xFFFFFFF8. If the MP comes up halted, the instruction cache fill does not take place until the first occurrence of an EINT3 interrupt to unhalt the MP.

absolute maximum ratings over specified temperature ranges (unless otherwise noted)†

Supply voltage range, V_{DD} (see Note 1) . . . . – 0.3 V to 4 V
Input voltage range, V_I . . . . – 0.3 V to 4 V
Output voltage range . . . . – 0.3 V to 4 V
Operating case temperature range, T_C . . . . 0°C to 85°C
Storage temperature range, T_{stg} . . . . – 55°C to 150°C

† Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: All voltage values are with respect to V_{SS}.

recommended operating conditions

		MIN	NOM	MAX	UNIT
V_{DD}	Supply voltage	3.135	3.3	3.465	V
V_{SS}	Supply voltage (see Note 2)		0		V
I_{OH}	High-level output current			– 400	µA
I_{OL}	Low-level output current			2	mA
T_C	Operating case temperature	0		85	°C

NOTE 2: In order to minimize noise on V_{SS}, care should be taken to provide a minimum inductance path between the V_{SS} pins and system ground.

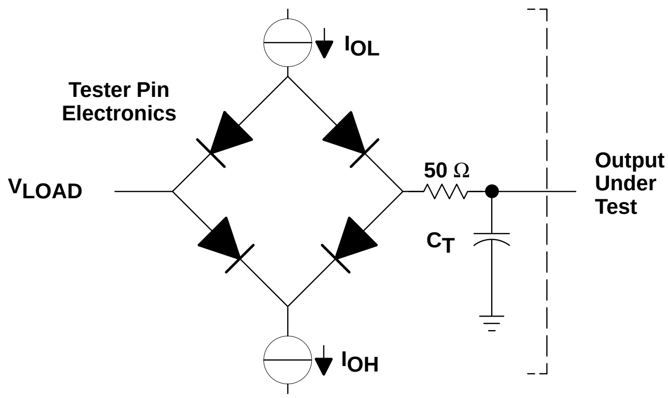
electrical characteristics over recommended ranges of supply voltage and operating case temperature (unless otherwise noted)

PARAMETER		TEST CONDITIONS‡		MIN	TYP§	MAX	UNIT
V_{IH}	High-level input voltage			2		V_{DD} + 0.3	V
V_{IL}	Low-level input voltage			– 0.3		0.8	V
V_{OH}	High-level output voltage	V_{DD} = MIN,	I_{OH} = MAX	2.6	¶		V
V_{OL}	Low-level output voltage	V_{DD} = MAX,	I_{OH} = MIN			0.6	V
I_O	Output current, leakage (high impedance) (except EMU0 and EMU1)	V_{DD} = MAX,	V_O = 2.8 V			20	µA
		V_{DD} = MAX,	V_O = 0.6 V			– 20	
I_I	Input current (except TCK, TDI, and TMS)	V_I = V_{SS} to V_{DD}				±20	µA
I_{DD}	Supply current (see Note 3)	V_{DD} = MAX,	60 MHz		1.2#	2.5#	A
		V_{DD} = MAX,	50 MHz		1.0#	2.3#	
		V_{DD} = MAX,	40 MHz		0.9#	1.9#	
C_i	Input capacitance				10		pF
C_o	Output capacitance				10		pF

‡ For conditions shown as MIN / MAX, use the appropriate value specified under the recommended operating conditions.
§ All typical values are at V_{DD} = 3.3 V, ambient air temperature = 25°C
¶ Typical steady-state V_{OH} will not exceed V_{DD}
# Parameter value is representative of revision 4.x and higher devices.

NOTE 3: Maximum supply current is derived from a test case that generates the theoretical maximum data flow using a worst case checkerboard data pattern on a sustained cycle by cycle basis. Actual maximum I_{DD} varies in real applications based on internal and external data flow and transitions. Typical supply current is derived from a test case which attempts to emulate typical use conditions of the on-chip processors with random data. Typical I_{DD} varies from application to application based on data flow and transitions and on-chip processor utilization.

PARAMETER MEASUREMENT INFORMATION



Where: I_{OL} = 2.0 mA (all outputs)
I_{OH} = 400 μA (all outputs)
V_{LOAD} = 1.5 V
C_T = 60 pF typical load circuit capacitance

Figure 115. Test Load Circuit

signal transition levels

TTL-output levels are driven to a minimum logic-high level of 2.4 V and to a maximum logic-low level of 0.6 V. Figure 116 shows the TTL-level outputs.

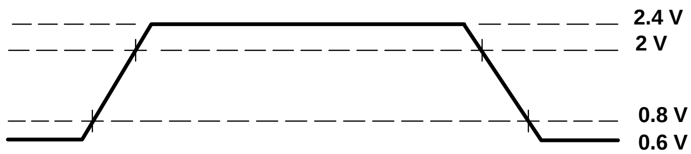


Figure 116. TTL-Level Outputs

TTL-output transition times are specified as follows:

- For a high-to-low transition, the level at which the output is said to be no longer high is 2 V, and the level at which the output is said to be low is 0.8 V.
- For a low-to-high transition, the level at which the output is said to be no longer low is 0.8 V, and the level at which the output is said to be high is 2 V.

Figure 117 shows the TTL-level inputs.

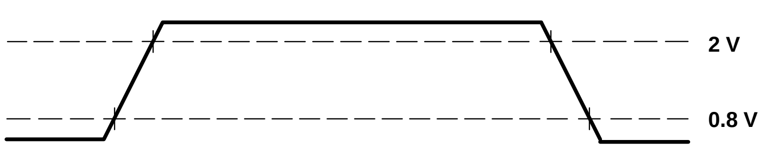


Figure 117. TTL-Level Inputs

TTL-compatible input transition times are specified as follows:

- For a high-to-low transition on an input signal, the level at which the input is said to be no longer high is 2 V, and the level at which the input is said to be low is 0.8 V.
- For a low-to-high transition on an input signal, the level at which the input is said to be no longer low is 0.8 V, and the level at which the input is said to be high is 2 V.

PARAMETER MEASUREMENT INFORMATION

timing parameter symbology

Timing parameter symbols used herein were created in accordance with JEDEC Standard 100-A. In order to shorten the symbols, some of the pin names and other related terminology have been abbreviated as follows:

A	A[31:0]	RDY	READY
CAS	CAS/DQM[7:0]	RST	RESET
CFG	AS[2:0], BS[1:0], CT[2:0], PS[3:0], UTIME	RTY	RETRY
CKI	CLKIN	REQ	REQ[1:0]
CKO	CLKOUT	RL	RL
CMP	RETRY, READY, FAULT	RR	READY, RETRY
D	D[63:0]	SCK	SCLK0, SCLK1
EIN	EINT1, EINT2, EINT3, or EINTx	TCK	TCK
EMU	EMU0, EMU1	TDI	TDI
FCK	FCLK0, FCLK1	TDO	TDO
HAK	HACK	TMS	TMS
HRQ	HREQ	TRS	TRST
LIN	LINT4	UTM	UTIME
MID	A[31:0], STATUS[5:0]	SI	HSYNC0, VSYNC0, CSYNC0, HSYNC1, VSYNC1, or CSYNC1
OUT	A[31:0], CAS/DQM[7:0], D[63:0], DBEN, DDIN, DSF, RAS, RL, STATUS[5:0], TRG/CAS, W	SY	HSYNC0, VSYNC0, CSYNC0/HBLNK0, CBLNK0/VBLNK0, HSYNC1, VSYNC1, CSYNC1/HBLNK1, CBLNK1/VBLNK1, CAREA0, or CAREA1
RAS	RAS	XPT	XPT[2:0] OR XPTx

Lowercase subscripts and their meanings are:

a	access time
c	cycle time (period)
d	delay time
h	hold time
su	setup time
t	transition time
w	pulse duration (width)

The following letters and symbols and their meanings are:

H	High
L	Low
V	Valid
Z	High impedance
X	Unknown, changing, or don’t care level

general notes on timing parameters

The period of the output clock (CLKOUT) is twice the period of the input clock (CLKIN), or $2 \times t_{c(CKI)}$. The half cycle time (t_H) that appears in the following tables is one-half of the output clock period, or equal to the input clock period, $t_{c(CKI)}$.

All output signals from the ’C80 (including CLKOUT) are derived from an internal clock such that all output transitions for a given half cycle occur with a minimum of skewing relative to each other.

The signal combinations shown in the following timing diagrams may not necessarily represent actual cycles. For actual cycle examples, refer to the appropriate cycle description section of this data sheet.

CLKIN timing requirements (see Figure 118)

NO.			’C80-40		’C80-50		’C80-60		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
1	$t_{c(CKI)}$	Period of CLKIN (t_H)	12.5		10		8.3		ns
2	$t_{w(CKIH)}$	Pulse duration of CLKIN high	4.8		4.2		3.9		ns
3	$t_{w(CKIL)}$	Pulse duration of CLKIN low	4.8		4.2		3.9		ns
4	$t_{t(CKI)}$	Transition time of CLKIN†		1.5		1.5		1.5	ns

† This parameter is verified by computer simulation and is not tested.

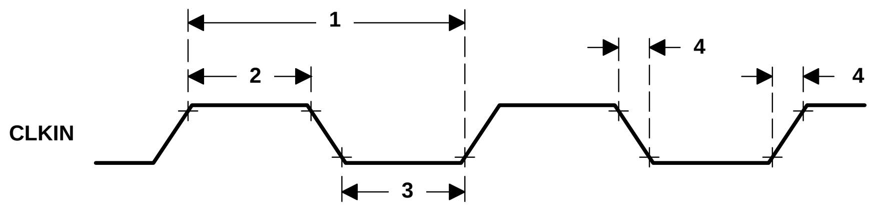


Figure 118. CLKIN Timing

local-bus switching characteristics over full operating range: CLKOUT‡(see Figure 119)

NO.	PARAMETER		’C80-40		’C80-50		’C80-60		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
5	$t_{c(CKO)}$	Period of CLKOUT	$2t_{c(CKI)}$§		$2t_{c(CKI)}$§		$2t_{c(CKI)}$§		ns
6	$t_{w(CKOH)}$	Pulse duration of CLKOUT high	$t_H - 5.5$		$t_H - 4.5$		$t_H - 3.7$		ns
7	$t_{w(CKOL)}$	Pulse duration of CLKOUT low	$t_H - 5.5$		$t_H - 4.5$		$t_H - 3.7$		ns
8	$t_{t(CKO)}$	Transition time of CLKOUT		2¶		2¶		2¶	ns

‡ The CLKOUT output has twice the period of CLKIN. No propagation delay or phase relationship to CLKIN is assured. Each state of a memory access begins on the falling edge of CLKOUT.
§ This is a functional minimum and is not tested. This parameter may also be specified as $2t_H$.
¶ This parameter is verified by computer simulation and is not tested.

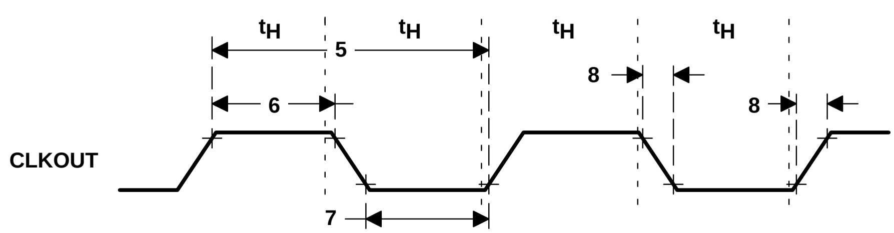


Figure 119. CLKOUT Timing

device reset timing requirements (see Figure 120)

NO.				MIN	MAX	UNIT
9	$t_{w(RSTL)}$	Pulse duration, $\overline{RESET}$ low	Initial reset during power-up	$6t_h$		ns
			Reset during active operation	$6t_h$		ns
10	$t_{su(HRQL\text{-}RSTH)}$	Setup time of $\overline{HREQ}$ low to $\overline{RESET}$ high to configure self-bootstrap mode		$4t_h$		ns
11	$t_{h(RSTH\text{-}HRQL)}$	Hold time, $\overline{HREQ}$ low after $\overline{RESET}$ high to configure self-bootstrap mode		0		ns
12	$t_{su(UTML\text{-}RSTH)}$	Setup time of $\overline{UTIME}$ low to $\overline{RESET}$ high to configure big-endian operation		$4t_h$		ns
13	$t_{h(RSTH\text{-}UTML)}$	Hold time, $\overline{UTIME}$ low after $\overline{RESET}$ high to configure big-endian operation		0		ns

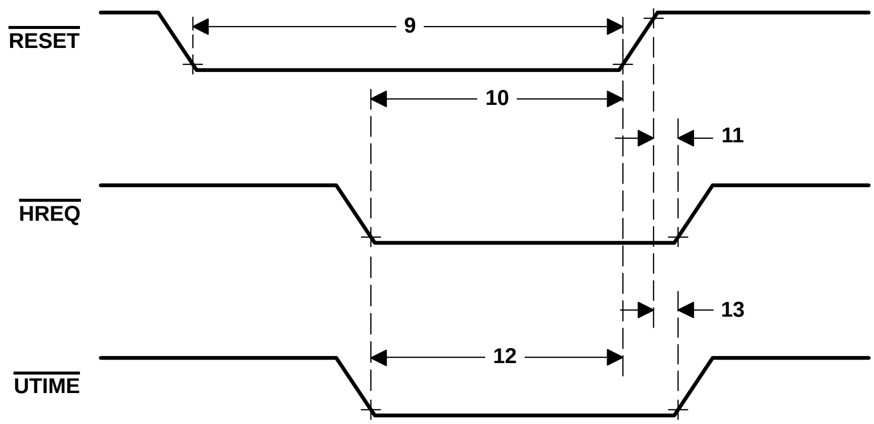


Figure 120. Device-Reset Timing

local bus timing requirements: cycle configuration inputs (see Figure 121)

The cycle configuration inputs are sampled at the beginning of each row access during the r2 state. The inputs typically are generated by a static decode of the A[31:0] and STATUS[5:0] outputs.

NO.			MIN	MAX	UNIT
14	$t_{su(CFGV\text{-}CKOH)}$	Setup time, AS, BS, CT, PS, and $\overline{UTIME}$ valid to CLKOUT no longer low	8		ns
15	$t_{h(CKOH\text{-}CFGV)}$	Hold time, AS, BS, CT, PS, and $\overline{UTIME}$ valid after CLKOUT high	2		ns
16	$t_{a(MIDV\text{-}CFGV)}$	Access time, AS, BS, CT, PS, and $\overline{UTIME}$ valid after memory identification (A, STATUS) valid		$3t_H - 10$	ns

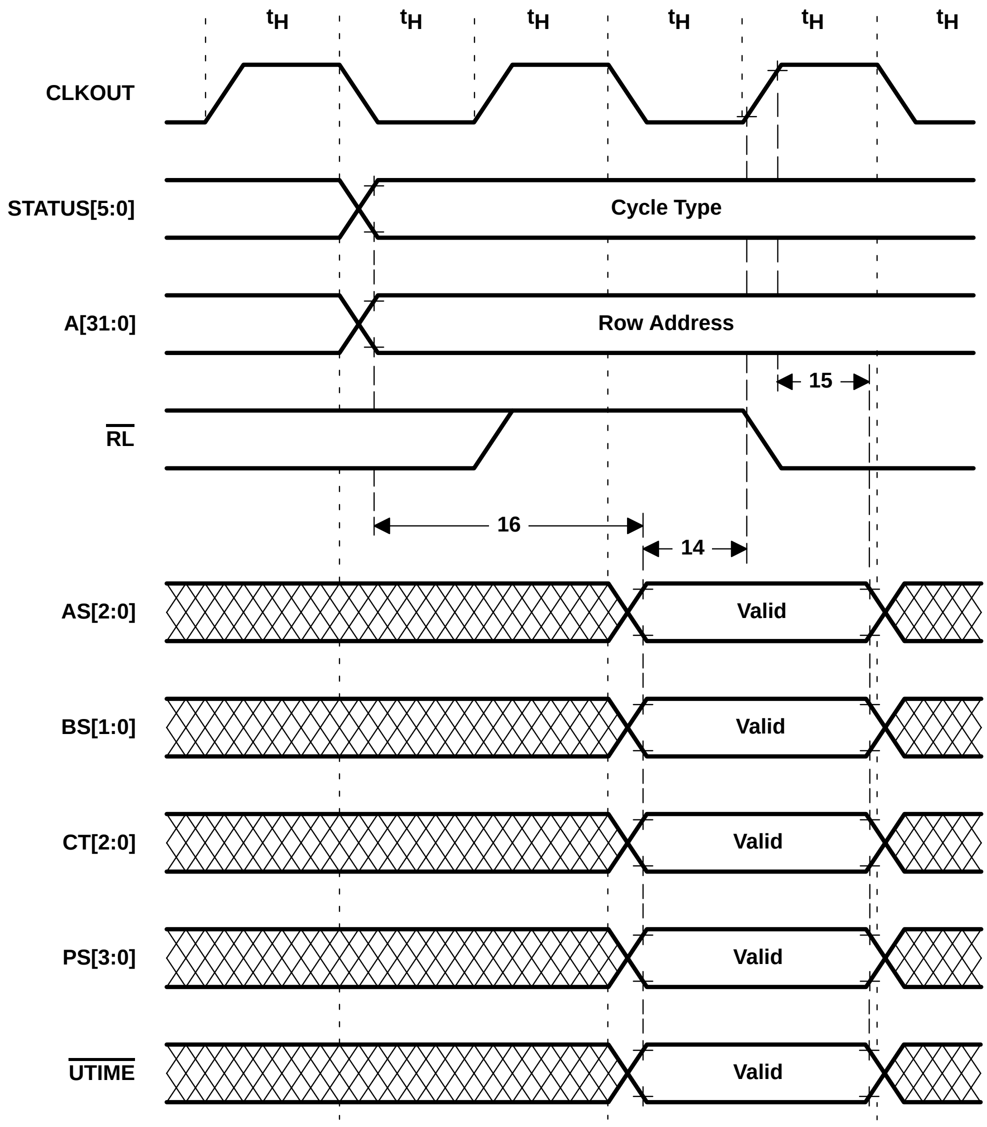


Figure 121. Local Bus Timing: Cycle Configuration Inputs

local bus timing: cycle completion inputs (see Figure 122 and Figure 123)

The cycle completion inputs are sampled at the beginning of each row access at the start of the r3 state. The READY input is also sampled at the start of the r6 state and during each column access (2 and 3 cyc/col accesses only). The $\overline{\text{RETRY}}$ input is sampled on each CLKOUT falling edge following r3. The value *n* as used in the parameters represents the integral number of half cycles between the transitions of the two signals in question.

NO.				’C80-40		’C80-50		’C80-60		UNIT
				MIN	MAX	MIN	MAX	MIN	MAX	
17	$t_{a(MIDV-CMPV)}$	Access time, $\overline{\text{RETRY}}$, READY, $\overline{\text{FAULT}}$ valid after memory identification (A, STATUS) valid			nt_H–9		nt_H–8		nt_H–7	ns
18	$t_{su(CMPV-CKOL)}$	Setup time, $\overline{\text{RETRY}}$, READY, $\overline{\text{FAULT}}$ valid to CLKOUT no longer high		8.0		7.5		7.5		ns
19	$t_{h(CKOL-CMPV)}$	Hold time, $\overline{\text{RETRY}}$, READY, $\overline{\text{FAULT}}$ valid after CLKOUT low		1.2		1.2		1.2		ns
20	$t_{a(RASL-RRV)}$	Access time $\overline{\text{RETRY}}$, READY valid from $\overline{\text{RAS}}$ low			nt_H–8		nt_H–7.5		nt_H–7.5	ns
21	$t_{a(RLL-RRV)}$	Access time, $\overline{\text{RETRY}}$, READY valid from $\overline{\text{RL}}$ low			nt_H–8		nt_H–7.5		nt_H–7.5	ns
22	$t_{a(CASL-RRV)}$	Access time, READY valid from $\overline{\text{CAS}}$ low (non-usertime mode)	2 cyc/col accesses		t_H–13.5		t_H–12		t_H–12	ns
			3 cyc/col accesses		$2t_H$–9		$2t_H$–8		$2t_H$–7	

local bus timing: cycle completion inputs (continued)

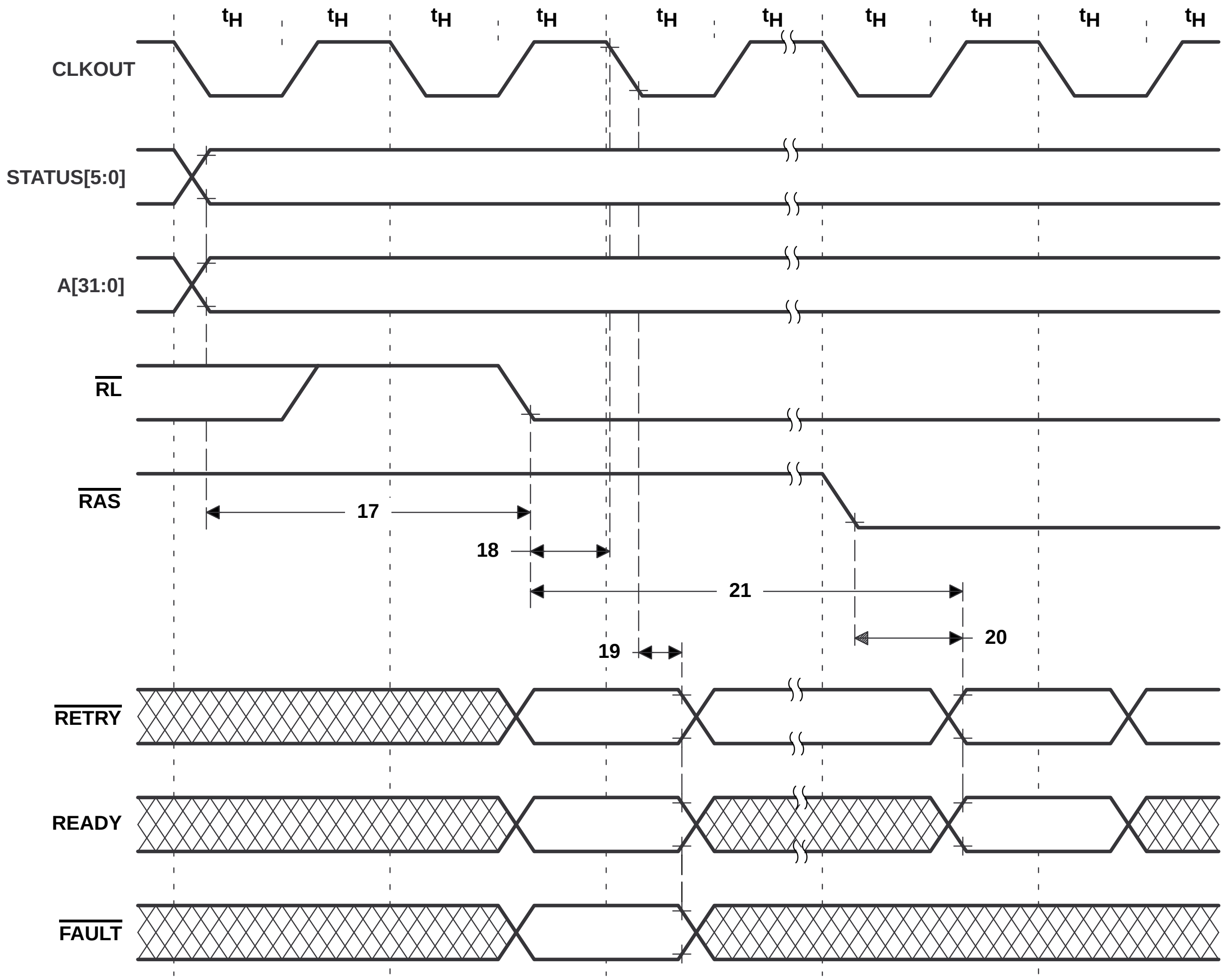


Figure 122. Local Bus Timing: Row-Time Cycle Completion Inputs

local bus timing: cycle completion inputs (continued)

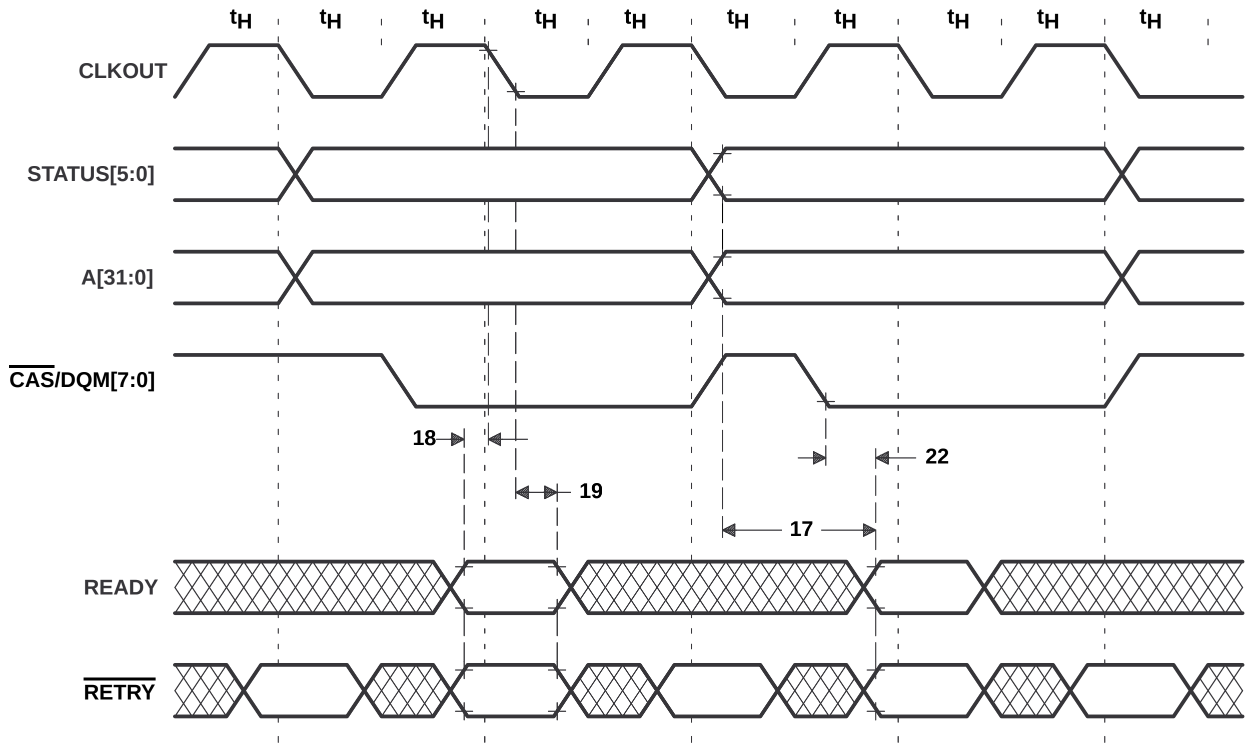


Figure 123. Local Bus Timing: Column-Time Cycle Completion Inputs

general output signal characteristics over full range of operating conditions

The following general timing parameters apply to all TMS320C80 output signals unless otherwise specifically given. The value *n* as used in the parameters represents the integral number of half cycles between the transitions of the two outputs in question. For timing purposes, outputs fall into one of three groups – the data bus (D[63:0]); the other output buses (A[31:0], STATUS[5:0], $\overline{\text{CAS}}$/DQM[7:0]); and non-bus outputs ($\overline{\text{DBEN}}$, $\overline{\text{DDIN}}$, DSF, $\overline{\text{RAS}}$, $\overline{\text{RL}}$, $\overline{\text{TRG}}$/$\overline{\text{CAS}}$, $\overline{\text{W}}$). When measuring output to output, the named group refers to the first output to transition (output A), and the second output (output B) refers to any output group (see Figure 124).

general output signal characteristics over full range of operating conditions† (continued)

NO.	PARAMETER		’C80-40		’C80-50		’C80-60		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
23	$t_{h(OUTV-CKOL)}$	Hold time, CLKOUT high after output valid D[63:0] A[31:0], STATUS[5:0], CAS/DQM[7:0‡ DBEN, DDIN, DSF, RAS, RL, TRG/CAS, W	 nt_H–7 nt_H–6.5 nt_H–5.5		 nt_H–5.3 nt_H–4.3 nt_H–3.9		 nt_H–5.3 nt_H–4.3 nt_H–3.9		ns
24	$t_{h(OUTV-CKOH)}$	Hold time, CLKOUT low after output valid D[63:0] A[31:0] STATUS[5:0], CAS/DQM[7:0]‡ DBEN, DDIN, DSF, RAS, RL, TRG/CAS, W	 nt_H–7 nt_H–6.5 nt_H–6.5 nt_H–5.5		 nt_H–5.5 nt_H–4.5 nt_H–4.5 nt_H–4.1		 nt_H–4 nt_H–4 nt_H–4.5 nt_H–4.1		ns
25	$t_{h(CKOL-OUTV)}$	Hold time, output valid after CLKOUT low	nt_H–5.5		nt_H–5		nt_H–5		ns
26	$t_{h(CKOH-OUTV)}$	Hold time, output valid after CLKOUT high	nt_H–5.5		nt_H–5		nt_H–4		ns
27	$t_{h(OUTV-OUTV)}$	Hold time, output valid after output valid D[63:0] A[31:0], STATUS[5:0], CAS/DQM[7:0]‡ DBEN, DDIN, DSF, RAS, RL, TRG/CAS, W	 nt_H–7 nt_H–6.5 nt_H–5.5		 nt_H–6.5 nt_H–5.5 nt_H–5		 nt_H–5.9 nt_H–5.5 nt_H–4.7		ns
28	$t_{d(CKOH-OUTV)}$	Delay time, CLKOUT no longer low to output valid D[63:0] A[31:0], STATUS[5:0], CAS/DQM[7:0]‡ DBEN, DDIN, DSF, RAS, RL, TRG/CAS, W		 nt_H+7 nt_H+6.5 nt_H+5.5		 nt_H+6.5 nt_H+5.5 nt_H+5		 nt_H+5.9 nt_H+5.5 nt_H+4.7	ns
29	$t_{d(CKOL-OUTV)}$	Delay time, CLKOUT no longer high to output valid D[63:0] A[31:0], STATUS[5:0], CAS/DQM[7:0]‡ DBEN, DDIN, DSF, RAS, RL, TRG/CAS, W		 nt_H+7 nt_H+6.5 nt_H+5.5		 nt_H+6.5 nt_H+5.5 nt_H+5		 nt_H+5.9 nt_H+5.5 nt_H+4.7	ns
30	$t_{d(OUTV-CKOH)}$	Delay time, output no longer valid to CLKOUT high		nt_H+5.5		nt_H+5		nt_H+5	ns
31	$t_{d(OUTV-CKOL)}$	Delay time, output no longer valid to CLKOUT low		nt_H+5.5		nt_H+5		nt_H+5	ns
32	$t_{d(OUTV-OUTV)}$	Delay time, output no longer valid to output valid D[63:0] A[31:0], STATUS[5:0], CAS/DQM[7:0]‡ DBEN, DDIN, DSF, RAS, RL, TRG/CAS, W		 nt_H+7 nt_H+6.5 nt_H+5.5		 nt_H+6.5 nt_H+5.5 nt_H+5		 nt_H+6.1 nt_H+5.5 nt_H+5	ns

† Tested across full voltage. Test temperature is selected by manufacturing test flow. Compliance across full temperature range is ensured by device characterization.
‡ Except for CAS/DQM[7:0] during non user-timed 2 cycle/column accesses

general output signal characteristics over full range of operating conditions† (continued)

NO.	PARAMETER		’C80-40		’C80-50		’C80-60		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
33	$t_{w(OUTV)}$	Pulse duration, output valid D[63:0]	nt_H–7		nt_H–6.5		nt_H–6.1		ns
		A[31:0], STATUS[5:0], $\overline{\text{CAS}}$/DQM[7:0]‡	nt_H–6.5		nt_H–5.5		nt_H–5.5		
		$\overline{\text{DBEN}}$, $\overline{\text{DDIN}}$, DSF, $\overline{\text{RAS}}$, $\overline{\text{RL}}$, $\overline{\text{TRG}}$/$\overline{\text{CAS}}$, $\overline{\text{W}}$	nt_H–5.5		nt_H–5		nt_H–5		

† Tested across full voltage. Test temperature is selected by manufacturing test flow. Compliance across full temperature range is ensured by device characterization.
‡ Except for $\overline{\text{CAS}}$/DQM[7:0] during non user-timed 2 cycle/column accesses

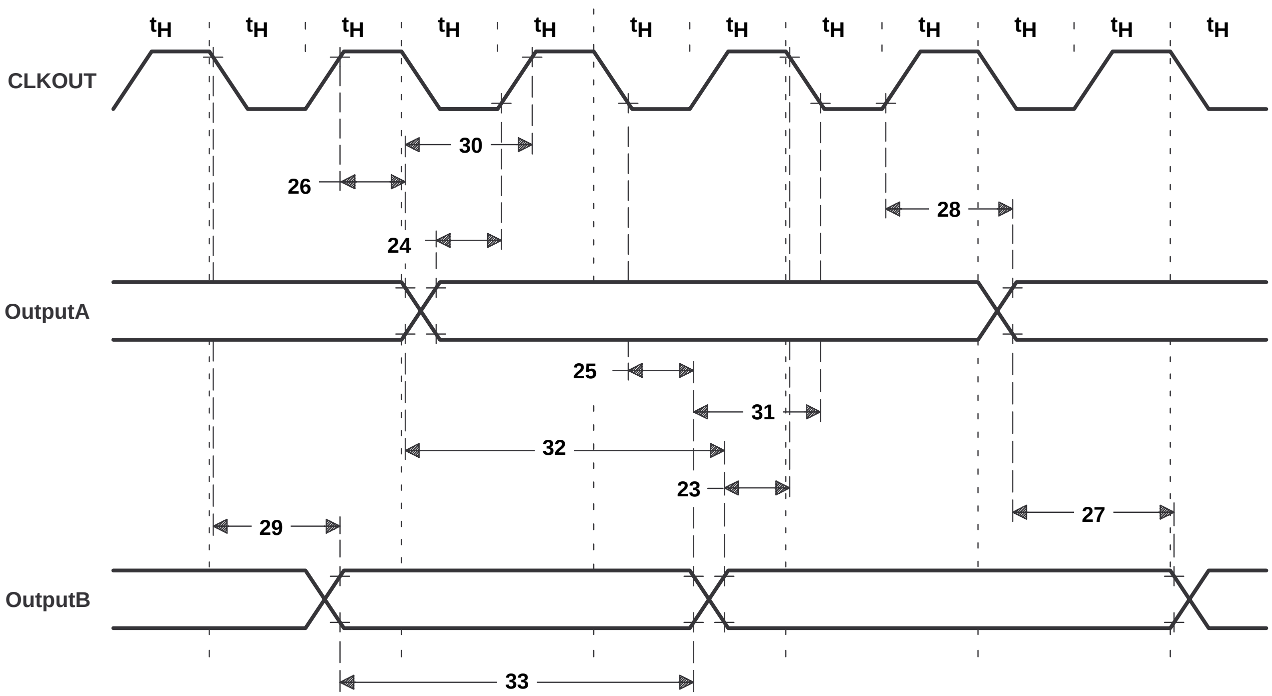


Figure 124. General Output-Signal Timing

data input timing

The following general timing parameters apply to the D[63:0] inputs unless otherwise specifically given. The value *n* as used in the parameters represents the integral number of half cycles between the transitions of the output and input in question (see Figure 125).

NO.	PARAMETER		’C80-40 MIN	’C80-40 MAX	’C80-50 MIN	’C80-50 MAX	’C80-60 MIN	’C80-60 MAX	UNIT
34	$t_{a(CKOH-DV)}$	Access time, CLKOUT high to D[63:0] valid		nt_H–8		nt_H–5.3		nt_H–4.0	ns
35	$t_{a(CKOL-DV)}$	Access time, CLKOUT low to D[63:0] valid		nt_H–8		nt_H–6.5		nt_H–6.5	ns
36	$t_{su(DV-CKOH)}$	Setup time, D[63:0] valid to CLKOUT no longer low	8		6.1		6.1		ns
37	$t_{su(DV-CKOL)}$	Setup time, D[63:0] valid to CLKOUT no longer high	8		6.1		6.1		ns
38	$t_{h(CKOL-DV)}$	Hold time, D[63:0] valid after CLKOUT low	2		2		2		ns
39	$t_{h(CKOH-DV)}$	Hold time, D[63:0] valid after CLKOUT high	2		2		2		ns
40	$t_{a(OUTV-DV)}$	Access time, output valid to D[63:0] inputs valid A[31:0], CAS/DQM[7:0]†, STATUS[5:0] DBEN, DDIN, DSF, RAS, RL, TRG/CAS, W		nt_H–9 nt_H–8		nt_H–7 nt_H–6.5		nt_H–7 nt_H–6.5	ns
41	$t_{h(OUTV-DV)}$‡	Hold time, D[63:0] valid after output valid RAS, CAS/DQM[7:0] A[31:0]	2 3		2 3		2 3		ns

† Except CAS/DQM[7:0] during non user-timed 2 cycle/column accesses
‡ Applies to RAS, CAS/DQM[7:0], and A[31:0] transitions that occur on CLKOUT edge coincident with input data sampling

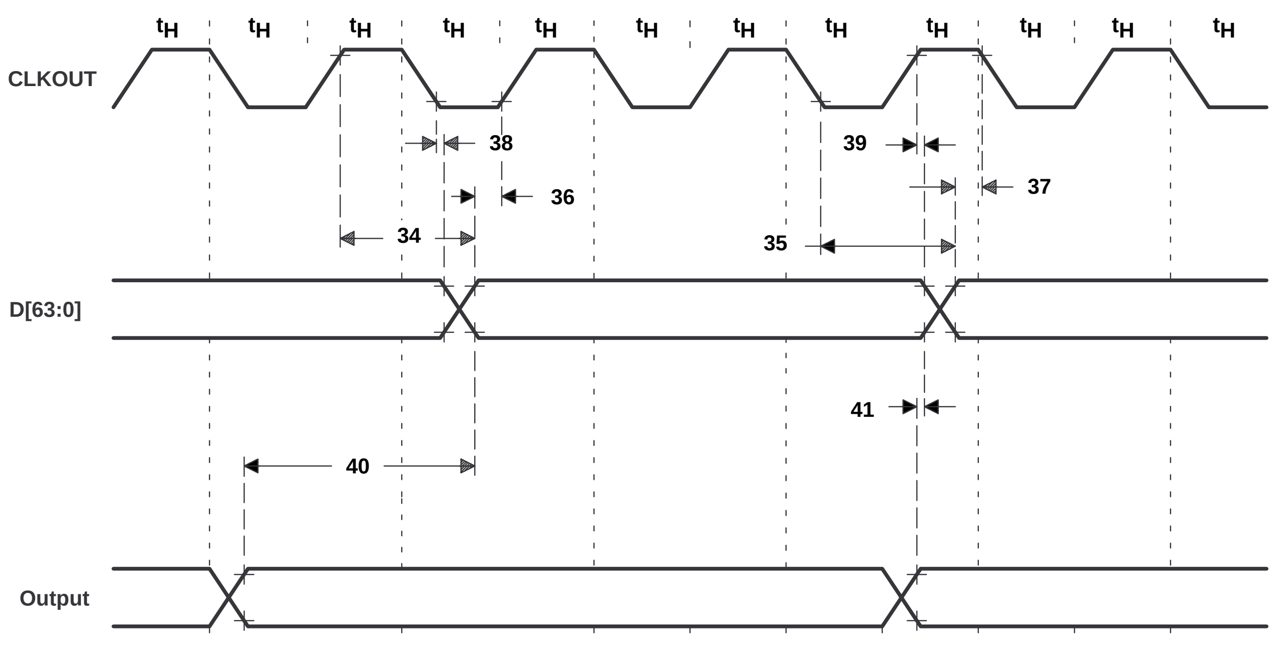


Figure 125. Data-Input Timing

local bus timing: 2 cycle/column $\overline{CAS}$ timing

These timing parameters apply to the $\overline{CAS}$/DQM[7:0] signals during 2 cycle per column memory accesses only. They should be used in place of the general output and data input timing parameters when the 2 cycle/column (non user-timed) cycle timing is selected (CT[2:0] inputs = 0b110). The value *n* as used in the parameters represents the integral number of half cycles between the transitions of the signals in question (see Figure 126).

NO.			’C80-40		’C80-50 ’C80-60		UNIT
			MIN	MAX	MIN	MAX	
42	$t_{w(CASH)}$	Pulse duration, $\overline{CAS}$/DQM high	t_H–2		t_H–2		ns
43	$t_{w(CASL)}$	Pulse duration, $\overline{CAS}$/DQM low	$3t_H$–11		$3t_H$–9.5		ns
44	$t_{h(OUTV\text{-}CASL)}$	Hold time, $\overline{CAS}$/DQM high after output valid D[63:0] A[31:0], STATUS[5:0] $\overline{DBEN}$, $\overline{DDIN}$, DSF, $\overline{RAS}$, $\overline{RL}$, $\overline{TRG}$/CAS, $\overline{W}$	 nt_H–5 nt_H–4.5 nt_H–3.5		 nt_H–4.5 nt_H–3.5 nt_H–3		ns
45	$t_{h(CASL\text{-}OUTV)}$	Hold time, output valid after $\overline{CAS}$/DQM low	nt_H–11		nt_H–9.5		ns
46	$t_{a(CASL\text{-}DV)}$	Access time, data valid from $\overline{CAS}$/DQM low		$3t_H$–12		$3t_H$–12	ns
47	$t_{h(CASH\text{-}DV)}$	Hold time, data valid after $\overline{CAS}$/DQM high	2		2		ns

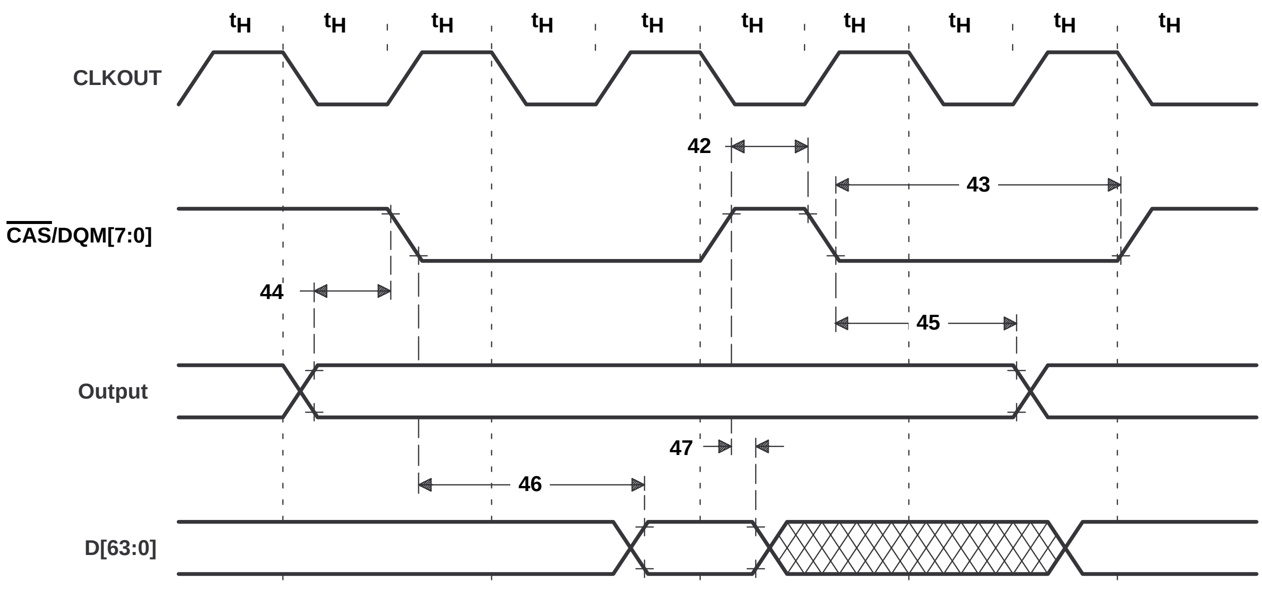


Figure 126. 2 Cycle / Column $\overline{CAS}$ Timing

external-interrupt timing

The following description defines the timing of the edge-triggered interrupts $\overline{EINT1}$ – $\overline{EINT3}$ and the level triggered interrupt $\overline{LINT4}$ (see Note 4). See Figure 127.

NO.			’C80-40 MIN	’C80-40 MAX	’C80-50 ’C80-60 MIN	’C80-50 ’C80-60 MAX	UNIT
48	$t_{w(EINL)}$	Pulse duration, $\overline{EINTx}$ low†	6		6		ns
49	$t_{su(EINH-CKOH)}$	Setup time, $\overline{EINTx}$ high before CLKOUT no longer low‡	11.5		9.5		ns
50	$t_{w(EINH)}$	Pulse duration, $\overline{EINTx}$ high†	6		6		ns
51	$t_{su(LINL-CKOL)}$	Setup time, $\overline{LINT4}$ low before CLKOUT no longer high‡	10		8		ns

† This parameter is assured by characterization and is not tested.
‡ This parameter must only be met to ensure that the interrupt is recognized on the indicated cycle.
NOTE 4: In order to assure recognition, $\overline{LINT4}$ must remain low until cleared by the interrupt service routine.

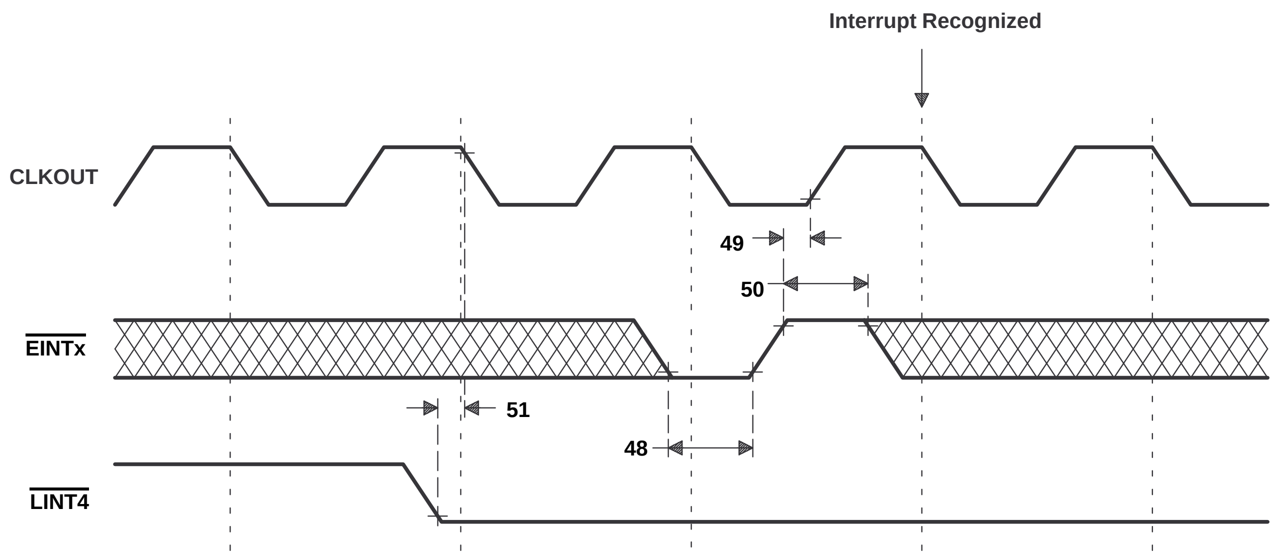


Figure 127. External-Interrupt Timing

XPT input timing

The following description defines the sampling of the $\overline{XPT[2:0]}$ inputs. The value encoded on the $\overline{XPT[2:0]}$ inputs is synchronized over multiple cycles to ensure that a stable value is present (see Figure 128 and Figure 129).

NO.			'C80-40		'C80-50 'C80-60		UNIT
			MIN	MAX	MIN	MAX	
52	$t_{w(XPTV)}$	Pulse duration, $\overline{XPTx}$ valid†	$12t_H$		$12t_H$		ns
53	$t_{su(XPTV-CKOH)}$	Setup time, $\overline{XPT[2:0]}$ valid before CLKOUT no longer low‡	13.5		12		ns
54	$t_{h(CKOH-XPTV)}$	Hold time, $\overline{XPT[2:0]}$ valid after CLKOUT high	5		5		ns
55	$t_{h(RLL-XPTV)}$	Hold time, $\overline{XPT[2:0]}$ valid after $\overline{RL}$ low§		$6t_H$		$6t_H$	ns

† This parameter is a functional minimum assured by logic and is not tested.
‡ This parameter must only be met to ensure that the XPT input is recognized on the indicated cycle.
§ This parameter must be met to ensure that a second XPT request does nor occur. This parameter is a functional maximum assured by logic and is not tested.

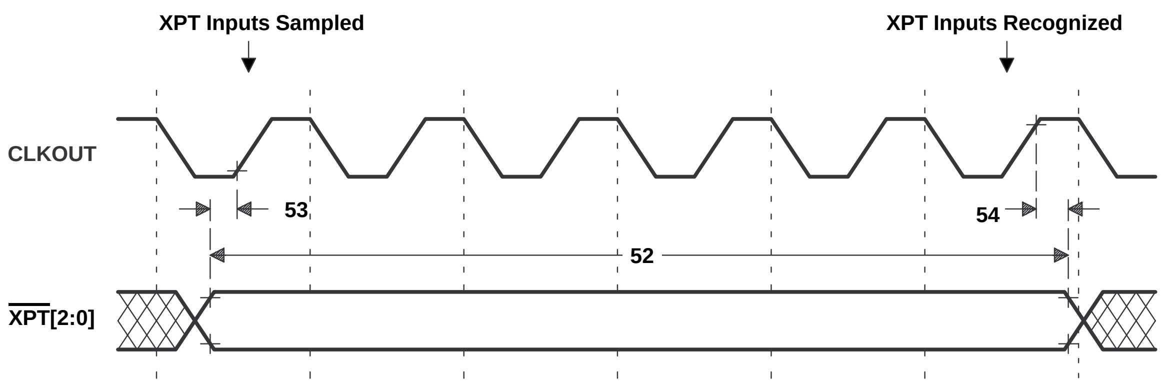


Figure 128. XPT Input Timing – XPT Recognition

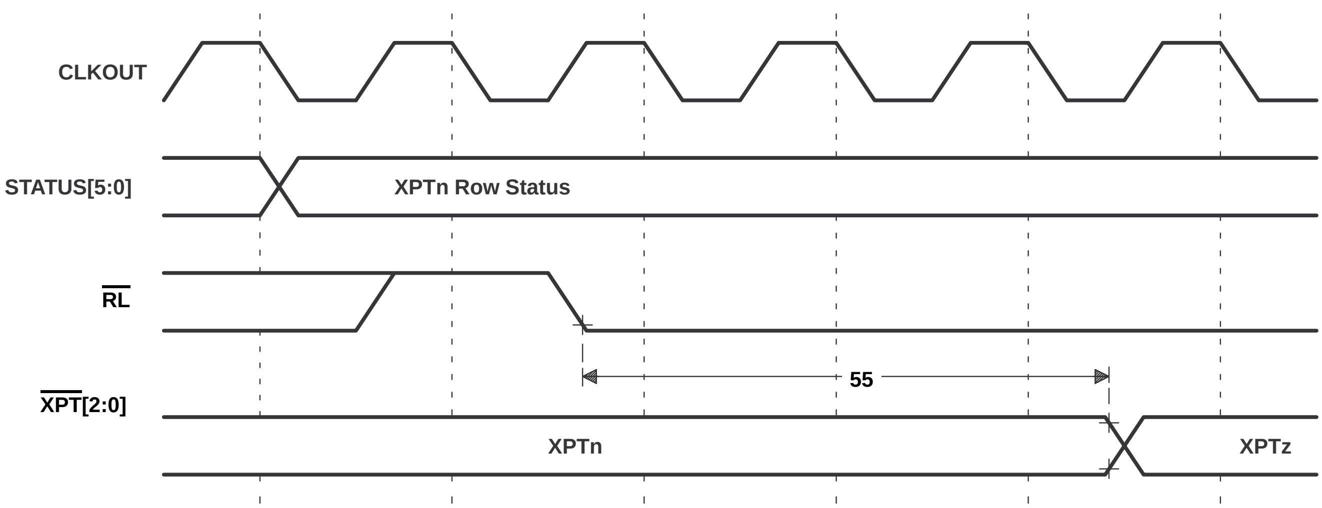


Figure 129. XPT Input Timing – XPT Service

host-interface timing (see Figure 130)

NO.				’C80-40		’C80-50		’C80-60		UNIT
				MIN	MAX	MIN	MAX	MIN	MAX	
56	$t_{su(REQV\text{-}CKOH)}$	Setup time, REQ1 – REQ0 valid to CLKOUT no longer low		t_H – 7		t_H – 7		t_H – 5.5		ns
57	$t_{h(CKOH\text{-}REQV)}$	Hold time, REQ1 – REQ0 valid after CLKOUT high		t_H – 7		t_H – 7		t_H – 5.5		ns
58	$t_{h(HRQL\text{-}HAKL)}$	Hold time for $\overline{\text{HACK}}$ high after $\overline{\text{HREQ}}$ goes low†		4t_H – 12		4t_H – 12		4t_H – 12		ns
59	$t_{d(HAKL\text{-}OUTZ)}$	Delay time, HACK low to output hi-Z‡	All signals except D[63:0]		0		0		0	ns
			D[63:0]		1		1		1	
60	$t_{d(HRQH\text{-}HAKH)}$	Delay time, $\overline{\text{HREQ}}$ high to $\overline{\text{HACK}}$ no longer low			10		10		10	ns
61	$t_{d(HAKH\text{-}OUTD)}$	Delay time, $\overline{\text{HACK}}$ high to outputs driven†		6t_H		6t_H		6t_H		
62	$t_{su(HRQL\text{-}CKOH)}$	Setup time, $\overline{\text{HREQ}}$ low to CLKOUT no longer low (see Note 5)		10.5		8.5		8.5		ns

† This parameter is a functional minimum assured by logic and is not tested.
‡ This parameter is assured by characterization and is not tested.
NOTE 5: Parameter must be met only to ensure $\overline{\text{HREQ}}$ recognition during the indicated clock cycle.

host-interface timing (see Figure 130) (continued)

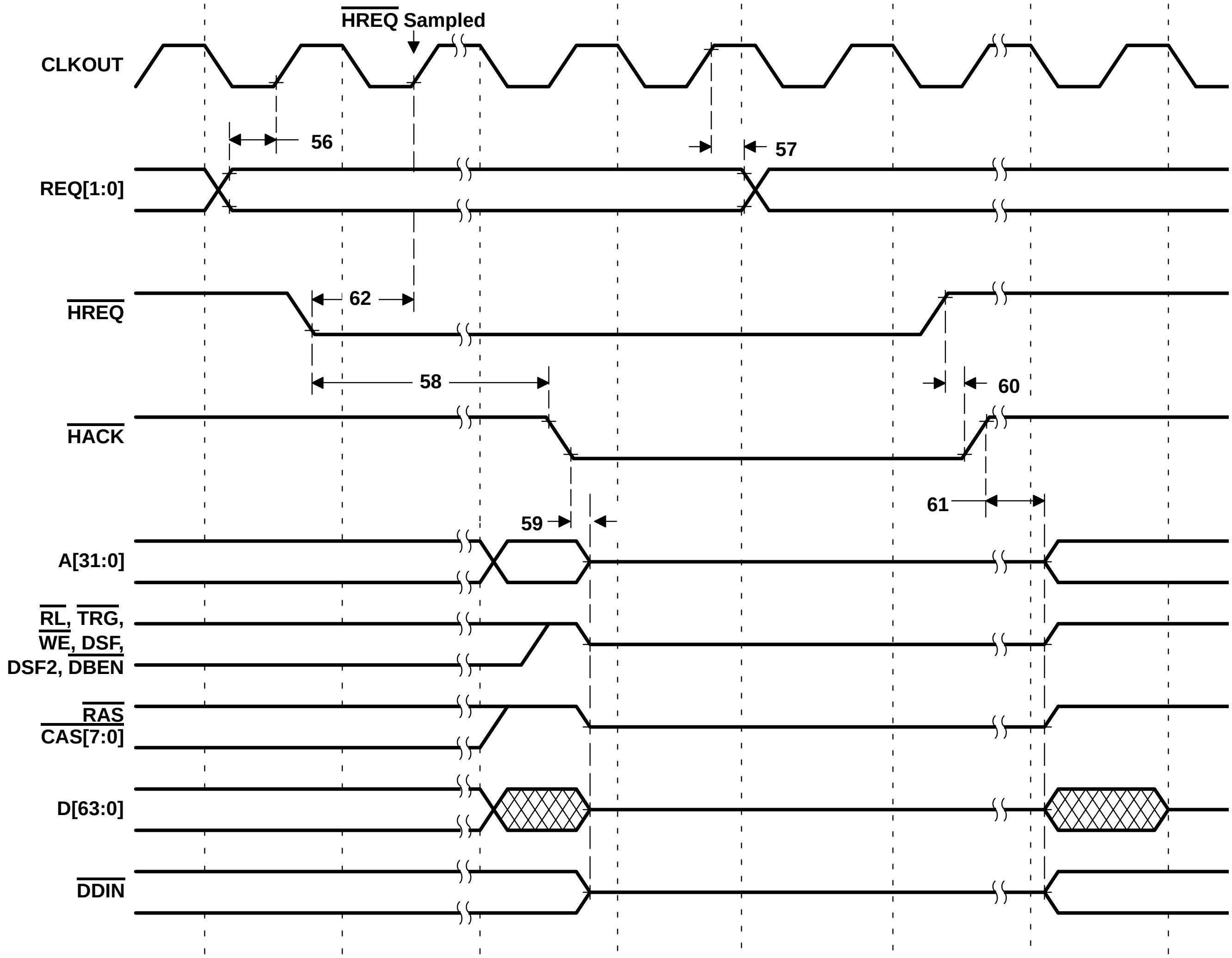


Figure 130. Host-Interface Timing

video interface timing: SCLK timing (see Figure 131)

NO.			MIN	MAX	UNIT
63	$t_{c(SCK)}$	SCLK period	13		ns
64	$t_{w(SCKH)}$	Pulse duration, SCLK high	5		ns
65	$t_{w(SCKL)}$	Pulse duration, SCLK low	5		ns
66	$t_{t(SCK)}$	Transition time, SCLK (rise and fall)†		2	ns

† This parameter is assured by simulation and is not tested.

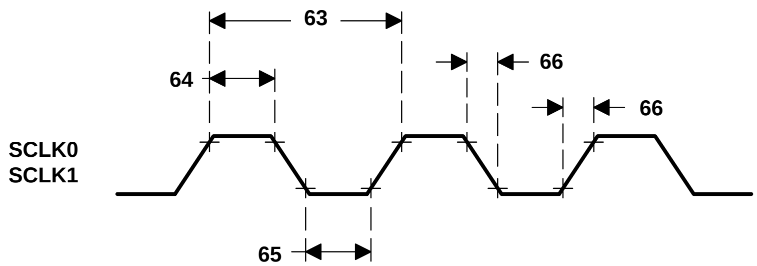


Figure 131. Video Interface Timing: SCLK Timing

video interface timing: FCLK input and video outputs (see Note 6 and Figure 132)

NO.			MIN	MAX	UNIT
67	$t_{c(FCK)}$	FCLK period	25		ns
68	$t_{w(FCKH)}$	Pulse duration, FCLK high	8		ns
69	$t_{w(FCKL)}$	Pulse duration, FCLK low	8		ns
70	$t_{t(FCK)}$	Transition time, FCLK (rise and fall)†		2	ns
71	$t_{h(FCKL-SYL)}$	Hold time, $\overline{HSYNC}$, $\overline{VSYNC}$, $\overline{CSYNC}$/$\overline{HBLNK}$, $\overline{CBLNK}$/$\overline{VBLNK}$, or CAREA high after FCLK low	0		ns
72	$t_{h(FCKL-SYH)}$	Hold time, $\overline{HSYNC}$, $\overline{VSYNC}$, $\overline{CSYNC}$/$\overline{HBLNK}$, $\overline{CBLNK}$/$\overline{VBLNK}$, or CAREA low after FCLK low	0		ns
73	$t_{d(FCKL-SYL)}$	Delay time, FCLK no longer high to $\overline{HSYNC}$, $\overline{VSYNC}$, $\overline{CSYNC}$/$\overline{HBLNK}$, $\overline{CBLNK}$/$\overline{VBLNK}$, or CAREA low		20	ns
74	$t_{d(FCKL-SYH)}$	Delay time, FCLK no longer high to $\overline{HSYNC}$, $\overline{VSYNC}$, $\overline{CSYNC}$/$\overline{HBLNK}$, $\overline{CBLNK}$/$\overline{VBLNK}$, or CAREA high		20	ns

† This parameter is assured by simulation and is not tested.

NOTE 6: Under certain circumstances these outputs also can transition asynchronously. These transitions occur when controller timing register values are modified by user programming. If the new register value forces the output to change states then this transition occurs without regard to FCLK inputs.

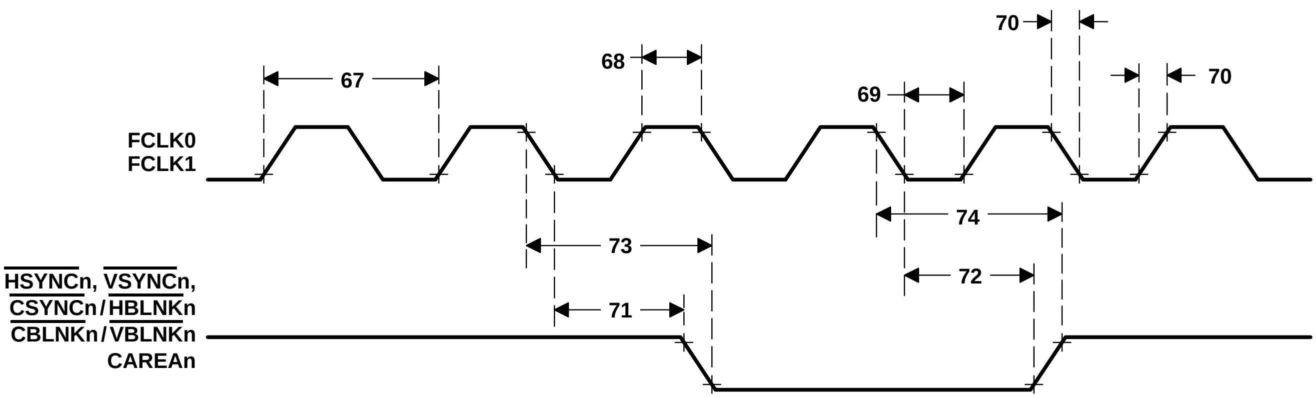


Figure 132. Video Interface Timing: FCLK Input and Video Outputs

video interface timing: external sync inputs

When configured as inputs, the $\overline{\mathrm{HSYNC}}$n, $\overline{\mathrm{VSYNC}}$n, and $\overline{\mathrm{CSYNC}}$n signals may be driven asynchronously. The following parameters apply only when the inputs are being generated synchronous to FCLKn in order to ensure recognition on a particular FLCKn edge (see Figure 133).

NO.			MIN	MAX	UNIT
75	$t_{su(SIL\text{-}FCKH)}$	Setup time, $\overline{\mathrm{HSYNC}}$, $\overline{\mathrm{VSYNC}}$, or $\overline{\mathrm{CSYNC}}$ low to FCLK no longer low†	5		ns
76	$t_{h(FCKH\text{-}SIL)}$	Hold time, $\overline{\mathrm{HSYNC}}$, $\overline{\mathrm{VSYNC}}$, or $\overline{\mathrm{CSYNC}}$ high after FCLK high‡	7		ns
77	$t_{su(SIH\text{-}FCKH)}$	Setup time, $\overline{\mathrm{HSYNC}}$, $\overline{\mathrm{VSYNC}}$, or $\overline{\mathrm{CSYNC}}$ high to FCLK no longer low§	5		ns
78	$t_{h(FCKH\text{-}SIH)}$	Hold time, $\overline{\mathrm{HSYNC}}$, $\overline{\mathrm{VSYNC}}$, or $\overline{\mathrm{CSYNC}}$ low after FCLK high¶	7		ns

† This parameter must be met only to ensure the input is recognized as low at FLCK edge B.
‡ This parameter must be met only to ensure the input is recognized as high at FLCK edge A.
§ This parameter must be met only to ensure the input is recognized as high at FLCK edge D.
¶ This parameter must be met only to ensure the input is recognized as low at FLCK edge C.

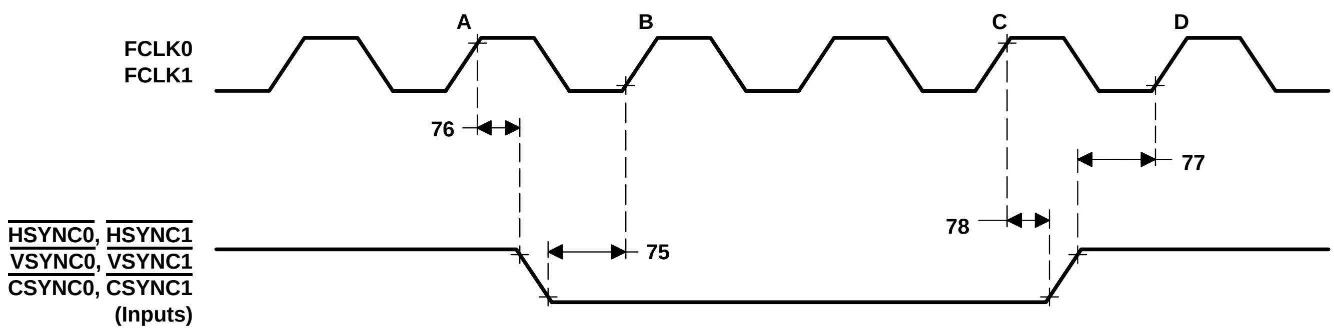
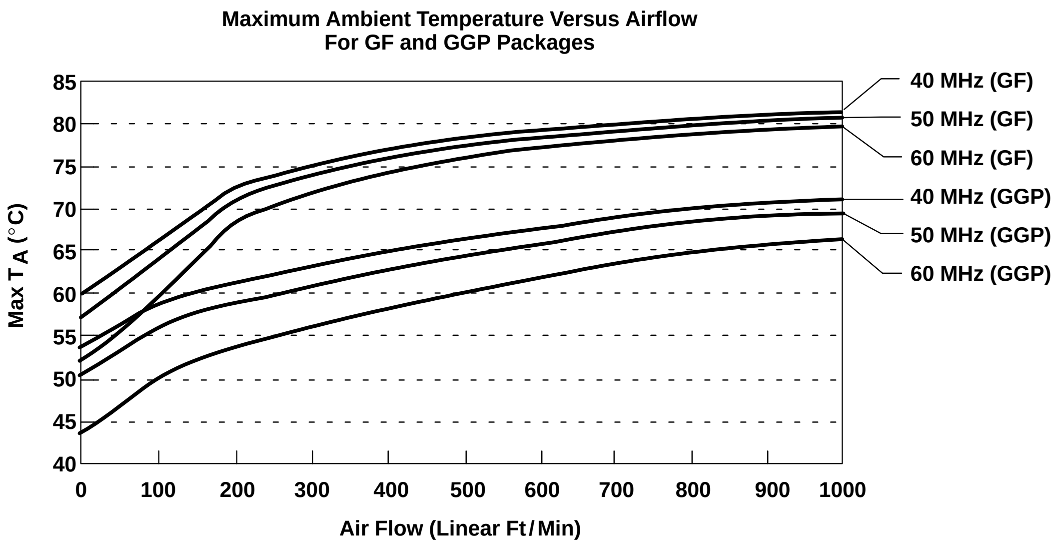


Figure 133. Video Interface Timing: External Sync Inputs

thermal resistance

Figure 134 illustrates the maximum ambient temperature allowed for various air flow rates across the TMS320C80 to ensure that the case temperature is kept below the maximum operating temperature (85°C) (see Note A). Values for the GF package include integral heat sink. Values for the GGP package are with no heat sink.



NOTE A: TMS320C80 power consumption is based on the “typical” values of I_{DD} measured at V_{DD} = 3.3 V. Power consumption varies by application based on TMS320C80 processor activity and I/O pin loadings. User must ensure that the case temperature (T_C) specifications are met when defining airflow and other thermal constraints of their system.

Figure 134. Airflow Requirements

emulator interface connection

The ’C80 supports emulation through a dedicated emulation port that is a superset of the IEEE Standard 1149.1 (JTAG) Standard. To support the ’C80 emulator, a target system must include a 14-pin header (2 rows of 7 pins) with the connections shown in Figure 135.

Signal	Pin	Pin	Signal
TMS	1	2	$\overline{TRST}$
TDI	3	4	GND
PD(+3.3V)	5	6	No pin (key)
TDO	7	8	GND
TCKRET	9	10	GND
TCK	11	12	GND
EMU0	13	14	EMU1

Pin Spacing: 0.100 in. (X,Y)
Pin Width: 0.025 in, square post
Pin Length: 0.235 in. nominal
(see Table 38)

Figure 135. Target System Header

Table 38. Target Connectors

XDS 510 SIGNAL	XDS 510 STATE	TARGET STATE	DESCRIPTION
TMS	O	I	Test-mode select†
TDI	O	I	Test-data input†
TDO	I	O	Test-data output†
TCK	O	I	Test clock – 10 MHz clock source from emulator. Can be used to drive system-test clock.†
$\overline{TRST}$	O	I	Test reset†
EMU0	I	I/O	Emulation pin 0
EMU1	I	I/O	Emulation pin 1
PD (3.3 V)	I	O	Presence detect. Indicates that the target is connected and powered up. Should be tied to + 3.3 V on target system.
TCKRET	I	O	Test clock return. Test clock input to the XDS 510 emulator. Can be buffered or unbuffered version of TCK.†

† IEEE Standard 1149.1.

For best results, the emulation header should be located as close as possible to the ’C80. If the distance exceeds six inches, the emulation signals should be buffered. See Figure 136.

emulator-interface connection (continued)

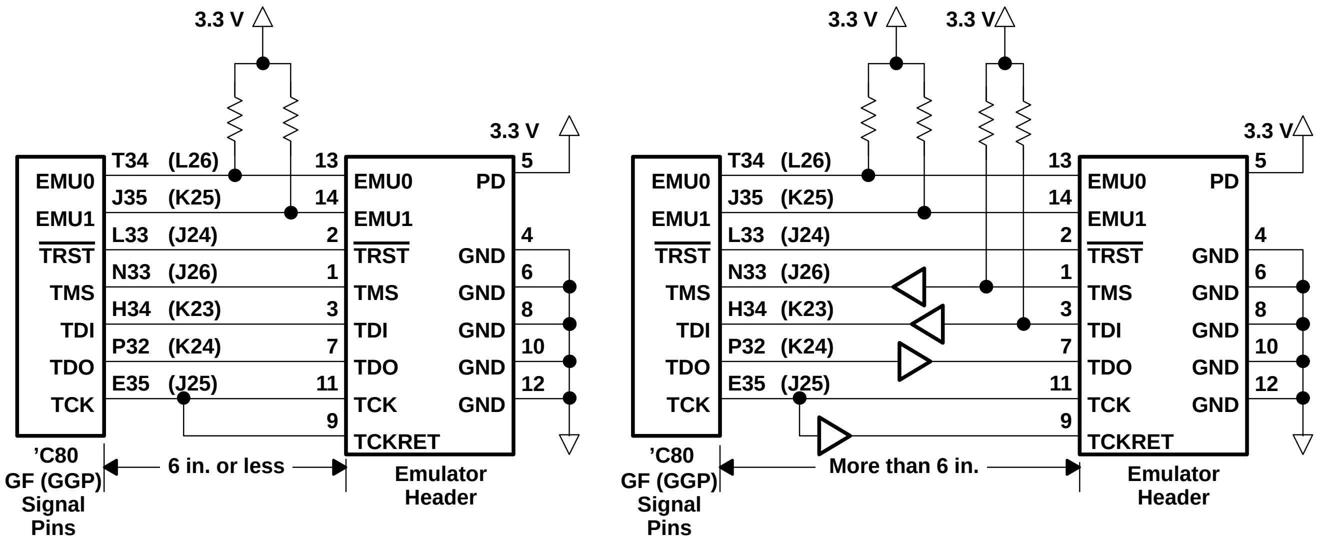


Figure 136. Emulation Header Connections – Emulator Driven Test Clock

The target system also can generate the test clock. This allows the user to:

- Set the test clock frequency to match the system requirements. (The emulator provides only a 10-MHz test clock.)
- Have other devices in the system that require a test clock when the emulator is not connected

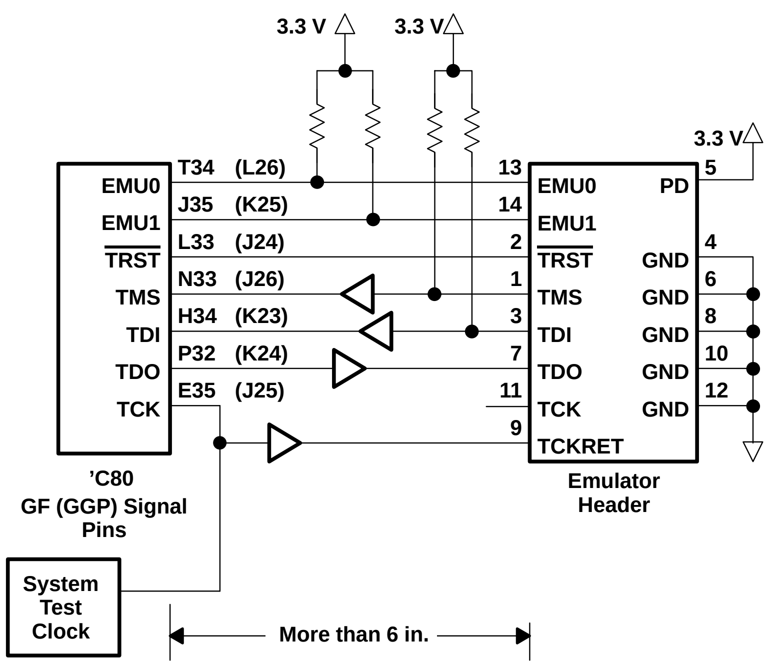


Figure 137. Emulation Header Connections – System Driven Test Clock

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77251–1443

emulator-interface connection (continued)

For multiprocessor applications, the following conditions are recommended:

- To reduce timing skew, buffer TMS, TDI, TDO, and TCK through the same physical package.
- If buffering is used, 4.7 kΩ resistors are recommended for TMS, TDI, and TCK which should be pulled high (3.3 V).
- Buffering EMU0 and EMU1 is highly recommended to provide isolation. The buffers need not be in the same physical package as TMS, TCK, TDI, or TDO. Pullups to 3.3 V are required and should provide a signal rise time of less than 10 µs. A 4.7 kΩ resistor is suggested for most applications.
- To ensure high quality signals, special printed wire board (PWB) routing and use of termination resistors may be required. The emulator provides fixed series termination (33 Ω) on TMS and TDI and optional parallel terminators (180 Ω pullup and 270 Ω pulldown) on TCKRET and TDO.

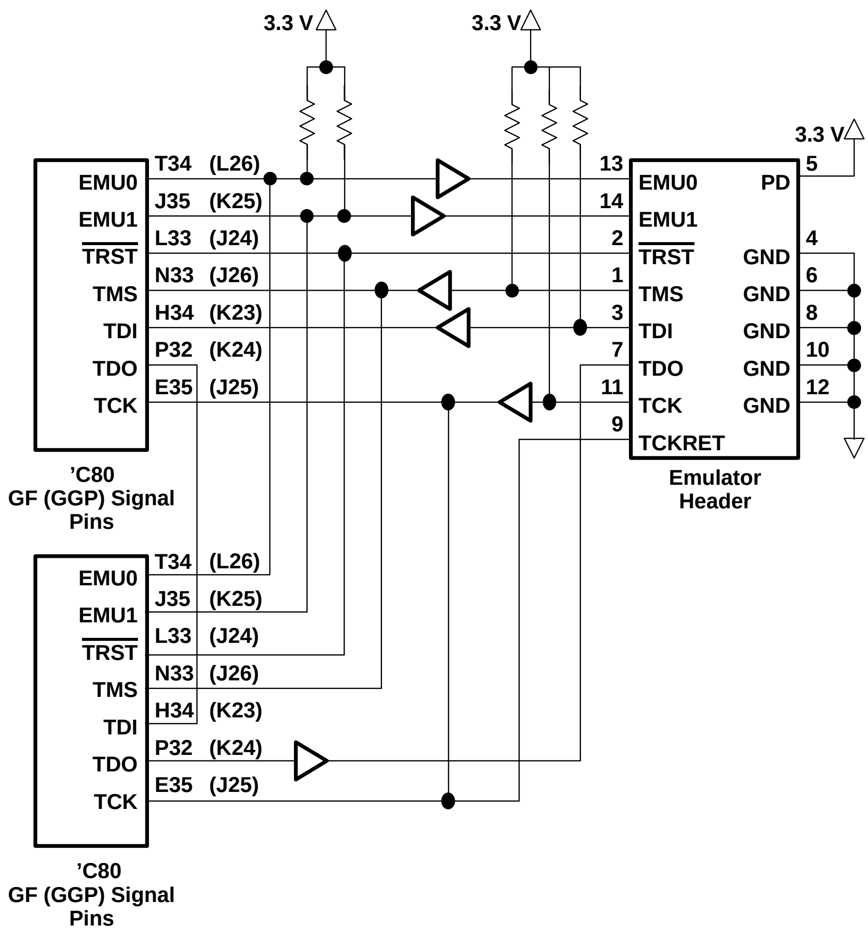
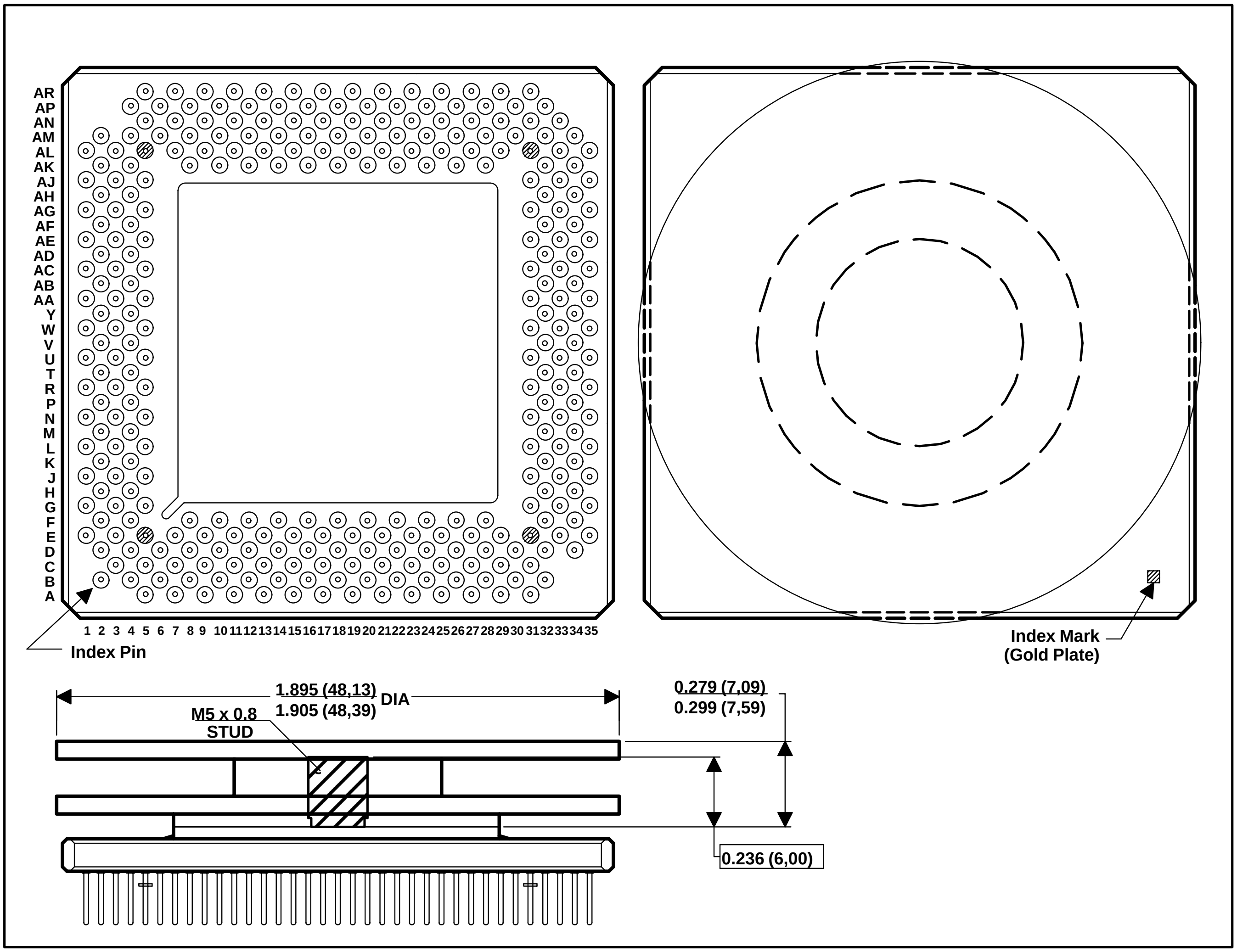


Figure 138. Emulation Header Connections – Multiprocessor Applications

GF package drawing



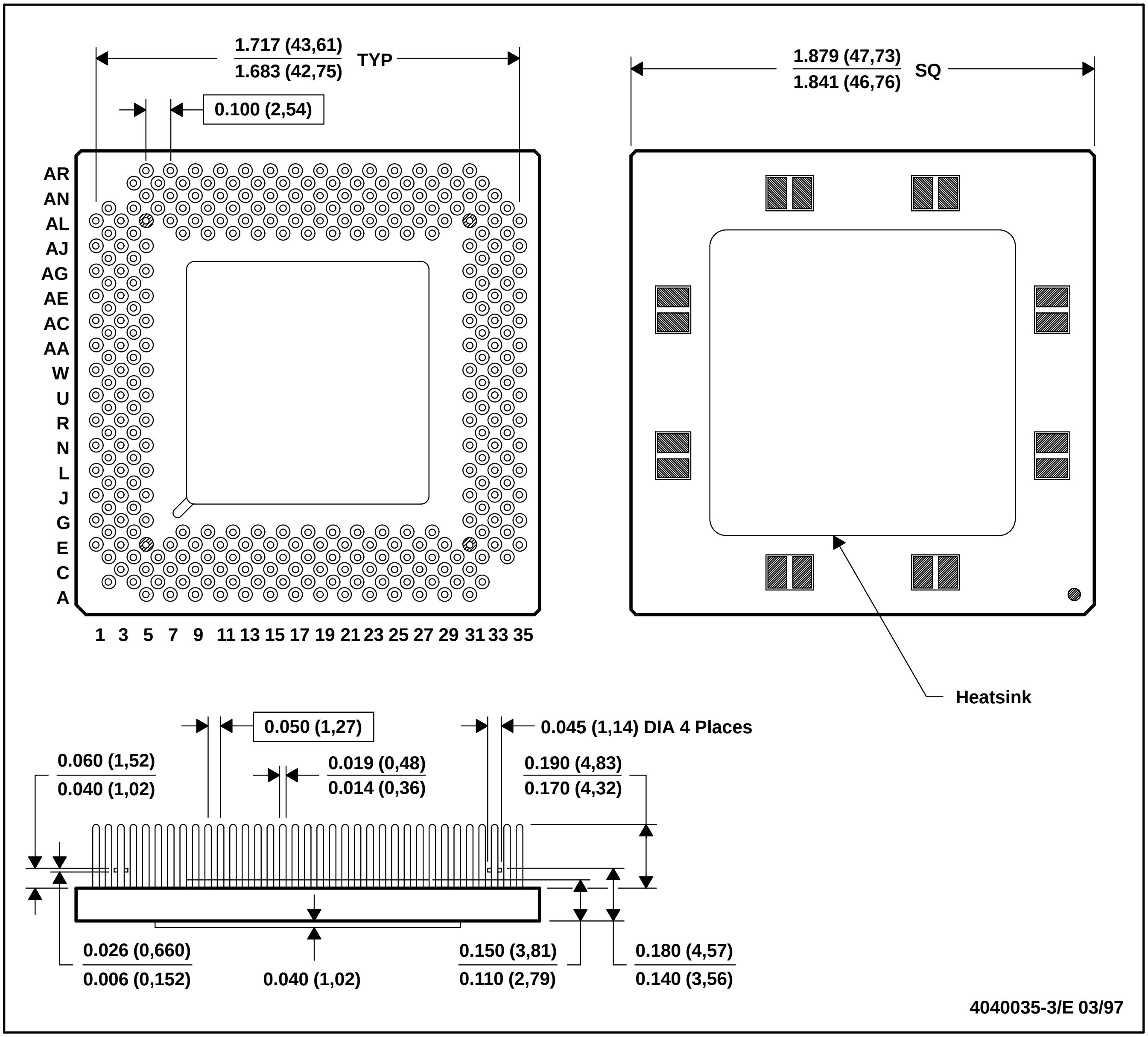
NOTES: A. Pins are located within 0,13 (0.005) radius of the true position relative to each other at maximum material condition and within 0,457 (0.018) radius of the center of the ceramic.

B. Dimensions do not include solder finish.

Figure 139. Assembled Package Drawing Showing Integral Heatsink

MECHANICAL DATA

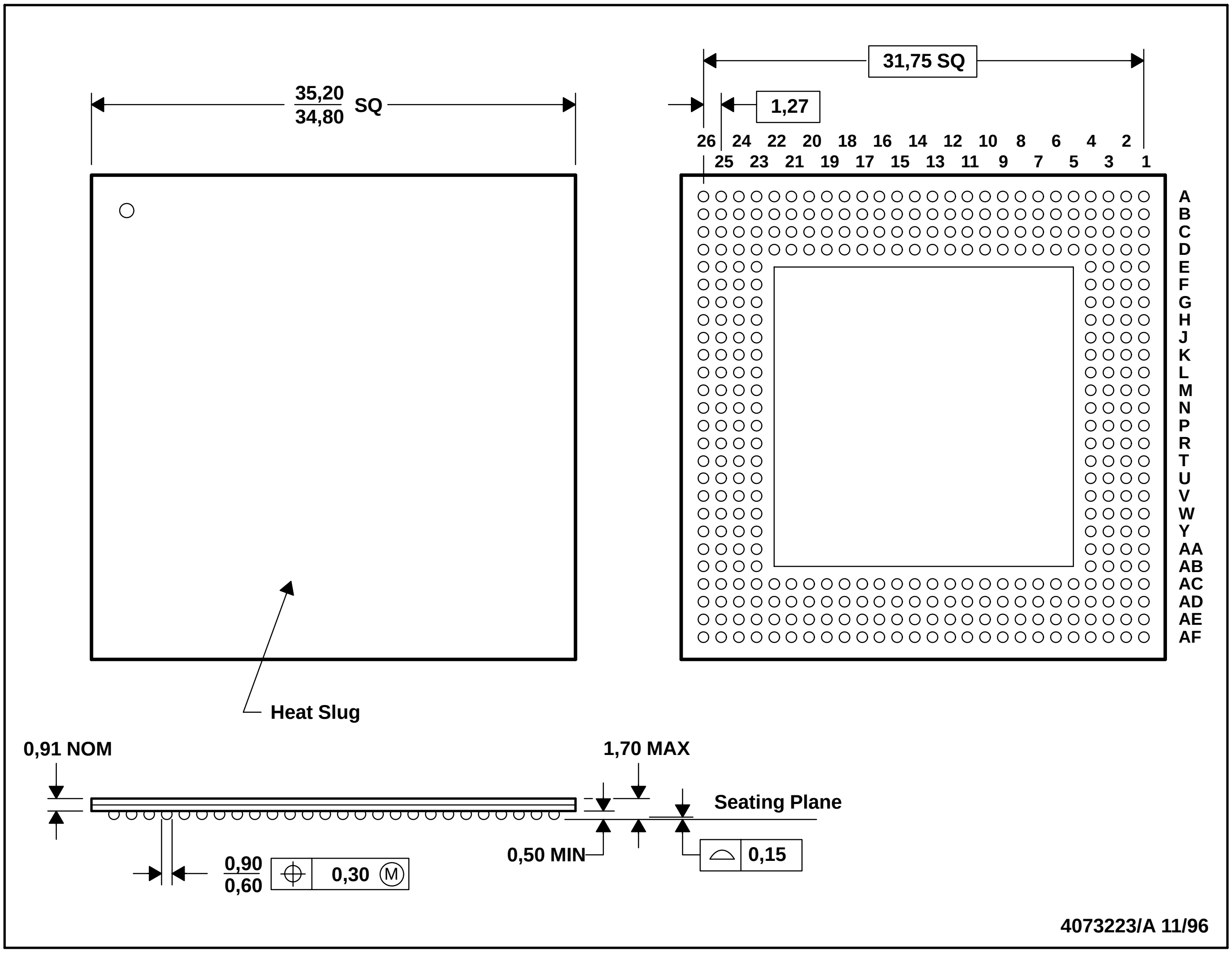
GF (S-CPGA-P305) **CERAMIC PIN GRID ARRAY PACKAGE**



NOTES: A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. Package thickness of 0.150 (3,81) / 0.110 (2,79) includes package body and lid, but does not include integral heatsink or attached features.

MECHANICAL DATA

GGP (S-PBGA-N352) **PLASTIC BALL GRID ARRAY (CAVITY DOWN) PACKAGE**



NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Thermally enhanced die down plastic package with top surface metal heat slug.

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.