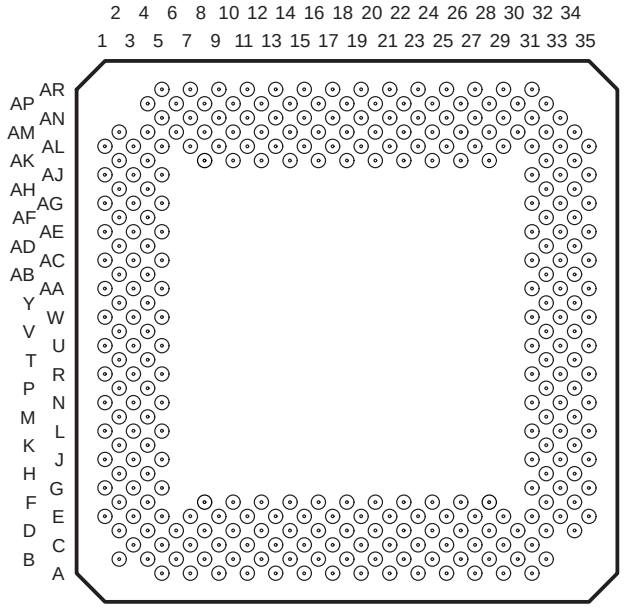
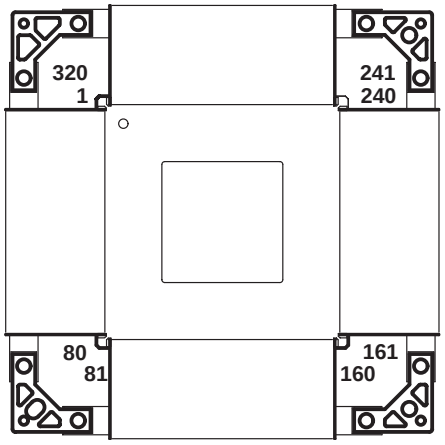
- **Single-Chip Parallel Multiple Instruction/Multiple Data (MIMD) Digital Signal Processor (DSP)**
- **More Than Two Billion RISC-Equivalent Operations per Second**
- **Master Processor (MP)**
  - **32-Bit Reduced Instruction Set Computing (RISC) Processor**
  - **IEEE-754 Floating-Point Capability**
  - **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 400 Megabytes per Second (MBps) 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 Video Random-Access Memory (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**
- **20-ns Cycle Time**
- **3.3-V Operation**
- **IEEE Standard 1149.1† Test Access Port (JTAG)**
- **Military Operating Temperature Range**
  - **–55°C to 125°C**

GF PACKAGE
(BOTTOM VIEW)



HFH PACKAGE
(TOP 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

Table of Contents

description

The SMJ320C80 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 100-MFLOPS (million floating-point operations per second) IEEE floating-point unit, four 32-bit parallel processing digital signal processors (DSPs), a transfer controller with up to 400-MBps 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 Pin Assignments – Numerical Listing

PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME
A5	CT1	C21	V_{DD}	E33	$\overline{HSYNC0}$	L5	V_{SS}
A7	V_{DD}	C23	$\overline{W}$	E35	TCK	L31	V_{SS}
A9	$\overline{HACK}$	C25	$\overline{DBEN}$	F2	V_{DD}	L33	$\overline{TRST}$
A11	V_{SS}	C27	V_{SS}	F4	V_{SS}	L35	$\overline{XPT1}$
A13	$\overline{CAS}$/DQM7	C29	CAREA0	F8	V_{DD}	M2	V_{DD}
A15	$\overline{CAS}$/DQM5	C31	$\overline{CBLNK0}$ / $\overline{VBLNK0}$	F10	V_{SS}	M4	V_{SS}
A17	V_{DD}	D2	$\overline{RETRY}$	F12	V_{DD}	M32	V_{SS}
A19	V_{SS}	D4	V_{DD}	F14	PS0	M34	V_{DD}
A21	$\overline{RAS}$	D6	V_{SS}	F16	V_{SS}	N1	V_{DD}
A23	DSF	D8	AS0	F18	CT2	N3	A8
A25	V_{SS}	D10	$\overline{UTIME}$	F20	V_{DD}	N5	V_{SS}
A27	SCLK1	D12	V_{SS}	F22	V_{SS}	N31	V_{SS}
A29	V_{DD}	D14	$\overline{RESET}$	F24	V_{DD}	N33	TMS
A31	$\overline{EINT1}$	D16	REQ0	F26	V_{SS}	N35	V_{DD}
B2	NC	D18	V_{SS}	F28	V_{DD}	P2	A4
B4	BS1	D20	$\overline{CAS}$/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	$\overline{XPT0}$
B10	REQ1	D26	CAREA1	G3	A2	R1	V_{SS}
B12	V_{DD}	D28	SCLK0	G5	A1	R3	V_{DD}
B14	$\overline{CAS}$/DQM6	D30	V_{SS}	G31	$\overline{EINT2}$	R5	V_{DD}
B16	$\overline{CAS}$/DQM3	D32	V_{DD}	G33	$\overline{CBLNK1}$ / $\overline{VBLNK1}$	R31	V_{DD}
B18	V_{DD}	D34	$\overline{VSYNC0}$	G35	V_{DD}	R33	V_{DD}
B20	$\overline{CAS}$/DQM1	E1	AS1	H2	STATUS0	R35	V_{SS}
B22	$\overline{TRG}$/$\overline{CAS}$	E3	$\overline{FAULT}$	H4	A3	T2	A5
B24	V_{DD}	E5	V_{SS}	H32	$\overline{CSYNC1}$ / $\overline{HBLNK1}$	T4	A13
B26	$\overline{DDIN}$	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	$\overline{CSYNC0}$ / $\overline{HBLNK0}$	E13	V_{SS}	J5	V_{DD}	U3	A10
C3	V_{SS}	E15	$\overline{HREQ}$	J31	V_{DD}	U5	PS3
C5	STATUS3	E17	$\overline{CAS}$/DQM4	J33	V_{SS}	U31	NC
C7	AS2	E19	$\overline{RL}$	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	$\overline{VSYNC1}$	V4	V_{SS}
C15	V_{DD}	E27	$\overline{LINT4}$	K34	$\overline{HSYNC1}$	V32	V_{SS}
C17	CLKIN	E29	$\overline{EINT3}$	L1	A0	V34	V_{DD}
C19	$\overline{CAS}$/DQM2	E31	V_{SS}	L3	A7	W1	A11

GF Pin Assignments – Numerical Listing (Continued)

PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN 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	NC	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 Pin Assignments – Alphabetical Listing

PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER
A0	L1	$\overline{\text{CAS}}$/DQM1	B20	D24	AL21	$\overline{\text{DBEN}}$	C25
A1	G5	$\overline{\text{CAS}}$/DQM2	C19	D25	AM22	$\overline{\text{DDIN}}$	B26
A2	G3	$\overline{\text{CAS}}$/DQM3	B16	D26	AP20	DSF	A23
A3	H4	$\overline{\text{CAS}}$/DQM4	E17	D27	AK22	$\overline{\text{EINT1}}$	A31
A4	P2	$\overline{\text{CAS}}$/DQM5	A15	D28	AN23	$\overline{\text{EINT2}}$	G31
A5	T2	$\overline{\text{CAS}}$/DQM6	B14	D29	AL25	$\overline{\text{EINT3}}$	E29
A6	K4	$\overline{\text{CAS}}$/DQM7	A13	D30	AR21	EMU0	T34
A7	L3	$\overline{\text{CBLNK0}}$/$\overline{\text{VBLNK0}}$	C31	D31	AM26	EMU1	J35
A8	N3	$\overline{\text{CBLNK1}}$/$\overline{\text{VBLNK1}}$	G33	D32	AL27	$\overline{\text{FAULT}}$	E3
A9	P4	CLKIN	C17	D33	AM28	FCLK0	B28
A10	U3	CLKOUT	E25	D34	AP22	FCLK1	D22
A11	W1	$\overline{\text{CSYNC0}}$/$\overline{\text{HBLNK0}}$	B32	D35	AN29	$\overline{\text{HACK}}$	A9
A12	Y2	$\overline{\text{CSYNC1}}$/$\overline{\text{HBLNK1}}$	H32	D36	AR23	$\overline{\text{HREQ}}$	E15
A13	T4	CT0	C11	D37	AN25	$\overline{\text{HSYNC0}}$	E33
A14	AB2	CT1	A5	D38	AL29	$\overline{\text{HSYNC1}}$	K34
A15	AE1	CT2	F18	D39	AP26	$\overline{\text{LINT4}}$	E27
A16	AG1	D0	AR5	D40	AR27	NC	B2
A17	AF2	D1	AN7	D41	AP28	NC	U31
A18	W3	D2	AM8	D42	AJ33	NC	AK18
A19	Y4	D3	AL7	D43	AR31	PS0	F14
A20	AH2	D4	AL9	D44	AH32	PS1	B8
A21	AB4	D5	AP8	D45	AN31	PS2	C13
A22	AC3	D6	AM10	D46	AF32	PS3	U5
A23	AL1	D7	AR9	D47	AP32	$\overline{\text{RAS}}$	A21
A24	AM2	D8	AP10	D48	AL33	READY	E9
A25	AL3	D9	AN11	D49	AF34	REQ0	D16
A26	AE3	D10	AL11	D50	AM34	REQ1	B10
A27	AP4	D11	AR13	D51	AE33	$\overline{\text{RESET}}$	D14
A28	AF4	D12	AN13	D52	AC33	$\overline{\text{RETRY}}$	D2
A29	AN5	D13	AP14	D53	AL35	$\overline{\text{RL}}$	E19
A30	AH4	D14	AM14	D54	AH34	SCLK0	D28
A31	AJ3	D15	AR15	D55	AB32	SCLK1	A27
AS0	D8	D16	AL15	D56	Y34	STATUS0	H2
AS1	E1	D17	AP16	D57	AG35	STATUS1	J1
AS2	C7	D18	AN17	D58	AE35	STATUS2	E7
BS0	E11	D19	AM16	D59	W33	STATUS3	C5
BS1	B4	D20	AL17	D60	AB34	STATUS4	K2
CAREA0	C29	D21	AL19	D61	U33		
CAREA1	D26	D22	AN19	D62	T32		
$\overline{\text{CAS}}$/DQM0	D20	D23	AM20	D63	W35		

GF Pin Assignments – Alphabetical Listing (Continued)

PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER
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
$\overline{UTIME}$	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

HFH Pin Assignments – Numerical Listing

PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME
1	STATUS3	41	$\overline{CAS}$/DQM6	81	$\overline{LINT4}$	121	V_{DD}
2	V_{SS}	42	V_{SS}	82	$\overline{EINT3}$	122	D59
3	STATUS2	43	$\overline{CAS}$/DQM5	83	$\overline{EINT2}$	123	V_{SS}
4	STATUS1	44	V_{DD}	84	$\overline{EINT1}$	124	D58
5	V_{DD}	45	$\overline{CAS}$/DQM4	85	$\overline{CBLNK1}$ / $\overline{VBLNK1}$	125	V_{DD}
6	STATUS0	46	$\overline{CAS}$/DQM3	86	$\overline{CBLNK0}$ / $\overline{VBLNK0}$	126	D57
7	AS2	47	CT2	87	V_{SS}	127	$\overline{XPT2}$
8	AS1	48	$\overline{CAS}$/DQM2	88	V_{SS}	128	V_{SS}
9	AS0	49	V_{SS}	89	$\overline{CSYNC1}$ / $\overline{HBLNK1}$	129	D56
10	$\overline{FAULT}$	50	$\overline{CAS}$/DQM1	90	V_{DD}	130	V_{DD}
11	READY	51	V_{DD}	91	V_{DD}	131	V_{DD}
12	$\overline{RETRY}$	52	$\overline{CAS}$/DQM0	92	$\overline{CSYNC0}$ / $\overline{HBLNK0}$	132	D55
13	$\overline{UTIME}$	53	$\overline{RL}$	93	VSYNC1	133	V_{SS}
14	BS1	54	$\overline{RAS}$	94	VSYNC0	134	D54
15	BS0	55	V_{SS}	95	V_{SS}	135	V_{DD}
16	CT1	56	V_{SS}	96	V_{SS}	136	D53
17	CT0	57	V_{SS}	97	$\overline{HSYNC1}$	137	V_{SS}
18	PS2	58	$\overline{TRG}$/$\overline{CAS}$	98	V_{DD}	138	V_{SS}
19	PS1	59	V_{DD}	99	V_{DD}	139	D52
20	PS0	60	FCLK1	100	V_{DD}	140	V_{DD}
21	V_{DD}	61	V_{DD}	101	$\overline{HSYNC0}$	141	D51
22	$\overline{RESET}$	62	V_{DD}	102	$\overline{TRST}$	142	D50
23	V_{SS}	63	$\overline{W}$	103	TCK	143	D49
24	$\overline{HREQ}$	64	STATUS5	104	TMS	144	V_{SS}
25	$\overline{HACK}$	65	V_{DD}	105	TDI	145	V_{SS}
26	V_{SS}	66	DSF	106	TDO	146	D48
27	V_{SS}	67	V_{SS}	107	EMU1	147	V_{DD}
28	REQ1	68	$\overline{DBEN}$	108	$\overline{XPT0}$	148	V_{DD}
29	REQ0	69	V_{DD}	109	$\overline{XPT1}$	149	V_{DD}
30	V_{DD}	70	$\overline{DDIN}$	110	V_{SS}	150	D47
31	V_{DD}	71	CLKOUT	111	V_{SS}	151	D46
32	V_{SS}	72	CAREA1	112	EMU0	152	D45
33	V_{DD}	73	V_{SS}	113	V_{DD}	153	V_{SS}
34	V_{SS}	74	SCLK1	114	D63	154	V_{SS}
35	V_{SS}	75	V_{DD}	115	D62	155	D44
36	CLKIN	76	FCLK0	116	V_{SS}	156	V_{DD}
37	V_{SS}	77	V_{SS}	117	D61	157	V_{DD}
38	$\overline{CAS}$/DQM7	78	SCLK0	118	V_{SS}	158	V_{DD}
39	V_{DD}	79	V_{DD}	119	D60	159	D43
40	V_{DD}	80	CAREA0	120	V_{DD}	160	D42

HFH Pin Assignments – Numerical Listing (Continued)

PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME
161	D41	201	D20	241	D0	281	V_{DD}
162	V_{SS}	202	V_{DD}	242	V_{DD}	282	V_{DD}
163	V_{SS}	203	V_{DD}	243	V_{DD}	283	V_{DD}
164	D40	204	D19	244	A31	284	A15
165	V_{DD}	205	V_{DD}	245	V_{SS}	285	PS3
166	D39	206	D18	246	A30	286	A14
167	D38	207	V_{SS}	247	A29	287	V_{SS}
168	D37	208	D17	248	V_{SS}	288	V_{DD}
169	V_{SS}	209	V_{SS}	249	V_{SS}	289	A13
170	D36	210	V_{SS}	250	A28	290	V_{SS}
171	V_{SS}	211	D16	251	V_{DD}	291	V_{SS}
172	V_{DD}	212	V_{DD}	252	V_{DD}	292	A12
173	D35	213	D15	253	V_{DD}	293	V_{DD}
174	D34	214	D14	254	A27	294	A11
175	D33	215	D13	255	A26	295	V_{SS}
176	V_{SS}	216	V_{SS}	256	A25	296	A10
177	D32	217	V_{SS}	257	V_{SS}	297	V_{DD}
178	V_{DD}	218	D12	258	V_{SS}	298	A9
179	V_{DD}	219	V_{DD}	259	V_{SS}	299	V_{SS}
180	D31	220	V_{DD}	260	A24	300	A8
181	D30	221	V_{DD}	261	V_{DD}	301	V_{DD}
182	D29	222	D11	262	V_{DD}	302	A7
183	V_{SS}	223	D10	263	V_{DD}	303	A6
184	V_{SS}	224	D9	264	A23	304	V_{SS}
185	V_{SS}	225	V_{SS}	265	A22	305	V_{SS}
186	D28	226	D8	266	V_{DD}	306	A5
187	V_{DD}	227	V_{DD}	267	A21	307	V_{SS}
188	V_{DD}	228	V_{DD}	268	V_{SS}	308	A4
189	D27	229	D7	269	V_{SS}	309	V_{DD}
190	D26	230	D6	270	A20	310	V_{DD}
191	D25	231	D5	271	V_{DD}	311	V_{DD}
192	V_{SS}	232	V_{SS}	272	V_{DD}	312	A3
193	D24	233	V_{SS}	273	A19	313	V_{DD}
194	V_{DD}	234	D4	274	V_{SS}	314	A2
195	V_{DD}	235	V_{DD}	275	A18	315	V_{SS}
196	D23	236	D3	276	A17	316	A1
197	D22	237	D2	277	V_{SS}	317	A0
198	V_{SS}	238	V_{SS}	278	V_{SS}	318	V_{DD}
199	D21	239	D1	279	V_{SS}	319	STATUS4
200	V_{SS}	240	V_{SS}	280	A16	320	V_{SS}

HFH Pin Assignments – Alphabetical Listing

PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER
A0	317	$\overline{\text{CAS}}$/DQM1	50	D30	181	$\overline{\text{DBEN}}$	68
A1	316	$\overline{\text{CAS}}$/DQM2	48	D31	180	$\overline{\text{DDIN}}$	70
A10	296	$\overline{\text{CAS}}$/DQM3	46	D32	177	DSF	66
A11	294	$\overline{\text{CAS}}$/DQM4	45	D33	175	$\overline{\text{EINT1}}$	84
A12	292	$\overline{\text{CAS}}$/DQM5	43	D34	174	$\overline{\text{EINT2}}$	83
A13	289	$\overline{\text{CAS}}$/DQM6	41	D35	173	$\overline{\text{EINT3}}$	82
A14	286	$\overline{\text{CAS}}$/DQM7	38	D36	170	EMU0	112
A15	284	$\overline{\text{CBLNK0}}$/$\overline{\text{VBLNK0}}$	86	D37	168	EMU1	107
A16	280	$\overline{\text{CBLNK1}}$/$\overline{\text{VBLNK1}}$	85	D38	167	$\overline{\text{FAULT}}$	10
A17	276	CLKIN	36	D39	166	FCLK0	76
A18	275	CLKOUT	71	D4	234	FCLK1	60
A19	273	$\overline{\text{CSYNC0}}$/$\overline{\text{HBLNK0}}$	92	D40	164	$\overline{\text{HACK}}$	25
A2	314	$\overline{\text{CSYNC1}}$/$\overline{\text{HBLNK1}}$	89	D41	161	$\overline{\text{HREQ}}$	24
A20	270	CT0	17	D42	160	$\overline{\text{HSYNC0}}$	101
A21	267	CT1	16	D43	159	$\overline{\text{HSYNC1}}$	97
A22	265	CT2	47	D44	155	$\overline{\text{LINT4}}$	81
A23	264	D0	241	D45	152	PS0	20
A24	260	D1	239	D46	151	PS1	19
A25	256	D10	223	D47	150	PS2	18
A26	255	D11	222	D48	146	PS3	285
A27	254	D12	218	D49	143	$\overline{\text{RAS}}$	54
A28	250	D13	215	D5	231	READY	11
A29	247	D14	214	D50	142	REQ0	29
A3	312	D15	213	D51	141	REQ1	28
A30	246	D16	211	D52	139	$\overline{\text{RESET}}$	22
A31	244	D17	208	D53	136	$\overline{\text{RETRY}}$	12
A4	308	D18	206	D54	134	$\overline{\text{RL}}$	53
A5	306	D19	204	D55	132	SCLK0	78
A6	303	D2	237	D56	129	SCLK1	74
A7	302	D20	201	D57	126	STATUS0	6
A8	300	D21	199	D58	124	STATUS1	4
A9	298	D22	197	D59	122	STATUS2	3
AS0	9	D23	196	D6	230	STATUS3	1
AS1	8	D24	193	D60	119	STATUS4	319
AS2	7	D25	191	D61	117	STATUS5	64
BS0	15	D26	190	D62	115	TCK	103
BS1	14	D27	189	D63	114	TDI	105
CAREA0	80	D28	186	D7	229		
CAREA1	72	D29	182	D8	226		
$\overline{\text{CAS}}$/DQM0	52	D3	236	D9	224		

HFH Pin Assignments – Alphabetical Listing (Continued)

PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER	PIN NAME	PIN NUMBER
TDO	106	V_{DD}	243	V_{SS}	110	V_{SS}	259
TMS	104	V_{DD}	251	V_{SS}	111	V_{SS}	26
$\overline{TRG}/\overline{CAS}$	58	V_{DD}	252	V_{SS}	116	V_{SS}	268
$\overline{TRST}$	102	V_{DD}	253	V_{SS}	118	V_{SS}	269
$\overline{UTIME}$	13	V_{DD}	261	V_{SS}	123	V_{SS}	27
V_{DD}	100	V_{DD}	262	V_{SS}	128	V_{SS}	274
V_{DD}	113	V_{DD}	263	V_{SS}	133	V_{SS}	277
V_{DD}	120	V_{DD}	266	V_{SS}	137	V_{SS}	278
V_{DD}	121	V_{DD}	271	V_{SS}	138	V_{SS}	279
V_{DD}	125	V_{DD}	272	V_{SS}	144	V_{SS}	287
V_{DD}	130	V_{DD}	281	V_{SS}	145	V_{SS}	290
V_{DD}	131	V_{DD}	282	V_{SS}	153	V_{SS}	291
V_{DD}	135	V_{DD}	283	V_{SS}	154	V_{SS}	295
V_{DD}	140	V_{DD}	288	V_{SS}	162	V_{SS}	299
V_{DD}	147	V_{DD}	293	V_{SS}	163	V_{SS}	304
V_{DD}	148	V_{DD}	297	V_{SS}	169	V_{SS}	305
V_{DD}	149	V_{DD}	30	V_{SS}	171	V_{SS}	307
V_{DD}	156	V_{DD}	301	V_{SS}	176	V_{SS}	315
V_{DD}	157	V_{DD}	309	V_{SS}	183	V_{SS}	320
V_{DD}	158	V_{DD}	310	V_{SS}	184	V_{SS}	34
V_{DD}	165	V_{DD}	311	V_{SS}	185	V_{SS}	35
V_{DD}	172	V_{DD}	313	V_{SS}	192	V_{SS}	37
V_{DD}	178	V_{DD}	318	V_{SS}	198	V_{SS}	42
V_{DD}	179	V_{DD}	39	V_{SS}	2	V_{SS}	49
V_{DD}	187	V_{DD}	40	V_{SS}	200	V_{SS}	55
V_{DD}	188	V_{DD}	33	V_{SS}	207	V_{SS}	56
V_{DD}	194	V_{DD}	44	V_{SS}	209	V_{SS}	57
V_{DD}	195	V_{DD}	5	V_{SS}	216	V_{SS}	67
V_{DD}	202	V_{DD}	51	V_{SS}	217	V_{SS}	73
V_{DD}	203	V_{DD}	59	V_{SS}	225	V_{SS}	77
V_{DD}	205	V_{DD}	61	V_{SS}	23	V_{SS}	87
V_{DD}	21	V_{DD}	62	V_{SS}	232	V_{SS}	88
V_{DD}	212	V_{DD}	65	V_{SS}	233	V_{SS}	95
V_{DD}	219	V_{DD}	69	V_{SS}	238	V_{SS}	96
V_{DD}	220	V_{DD}	75	V_{SS}	240	$\overline{VSYNC0}$	94
V_{DD}	221	V_{DD}	79	V_{SS}	245	$\overline{VSYNC1}$	93
V_{DD}	227	V_{DD}	90	V_{SS}	248	$\overline{W}$	63
V_{DD}	228	V_{DD}	91	V_{SS}	249	$\overline{XPT0}$	108
V_{DD}	235	V_{DD}	98	V_{SS}	257	$\overline{XPT1}$	109
V_{DD}	31	V_{DD}	99	V_{SS}	210	$\overline{XPT2}$	127
V_{DD}	242	V_{SS}	32	V_{SS}	258		

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 devices 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 bidirectional 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 synchronous dynamic random-access memories (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 synchronous graphics random-access memory (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
‡ This pin has an internal pullup and can be left unconnected during normal operation.
§ This pin has an internal pulldown and can be left unconnected during normal operation.
¶ For proper operation, all V_{DD} and V_{SS} pins must be connected externally.

Terminal Functions (Continued)

TERMINAL NAME	TYPE†	DESCRIPTION
HOST INTERFACE		
$\overline{\text{HACK}}$	O	Host acknowledge. The ’C80 drives $\overline{\text{HACK}}$ output low following an active $\overline{\text{HREQ}}$ to indicate that it has driven the local memory bus signals to the high-impedance state and is relinquishing the bus. $\overline{\text{HACK}}$ is driven high asynchronously following $\overline{\text{HREQ}}$ being detected inactive, and then the ’C80 resumes driving the bus.
$\overline{\text{HREQ}}$	I	Host request. An external device drives $\overline{\text{HREQ}}$ low to request ownership of the local memory bus. When $\overline{\text{HREQ}}$ is high, the ’C80 owns and drives the bus. $\overline{\text{HREQ}}$ is synchronized internally to the ’C80’s internal clock. Also, $\overline{\text{HREQ}}$ is used at reset to determine the power-up state of the MP. If $\overline{\text{HREQ}}$ is low at the rising edge of $\overline{\text{RESET}}$, the MP comes up running. If $\overline{\text{HREQ}}$ is high, the MP remains halted until the first interrupt occurrence on $\overline{\text{EINT3}}$.
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.
$\overline{\text{EINT1}}$, $\overline{\text{EINT2}}$, $\overline{\text{EINT3}}$	I	Edge-triggered interrupts. $\overline{\text{EINT1}}$, $\overline{\text{EINT2}}$ and $\overline{\text{EINT3}}$ allow external devices to interrupt the master processor (MP) on one of three interrupt levels ($\overline{\text{EINT1}}$ is the highest priority). The interrupts are rising-edge triggered. $\overline{\text{EINT3}}$ also serves as an unhalt signal. If the MP is powered-up halted, the first rising edge on $\overline{\text{EINT3}}$ causes the MP to unhalt and fetch its reset vector (the $\overline{\text{EINT3}}$ interrupt-pending bit is not set in this case).
$\overline{\text{LINT4}}$	I	Level-triggered interrupt. $\overline{\text{LINT4}}$ 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.
$\overline{\text{RESET}}$	I	Reset. $\overline{\text{RESET}}$ 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 $\overline{\text{RESET}}$, the MP reset mode and the ’C80’s operating endian mode are determined by the levels of $\overline{\text{HREQ}}$ and $\overline{\text{UTIME}}$ pins, respectively.
$\overline{\text{XPT2}}$–$\overline{\text{XPT0}}$	I	External packet transfer. $\overline{\text{XPT2}}$–$\overline{\text{XPT0}}$ 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.
$\overline{\text{TRST}}$§	I	Test reset. $\overline{\text{TRST}}$ 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 unconnected during normal operation.
§ This pin has an internal pulldown and can be left unconnected during normal operation.
¶ For proper operation, all V_{DD} and V_{SS} pins must be connected externally.

Terminal Functions (Continued)

TERMINAL NAME	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.
$\overline{\text{CBLNK0}}$ / $\overline{\text{VBLNK0}}$, $\overline{\text{CBLNK1}}$ / $\overline{\text{VBLNK1}}$	O	Composite blanking/vertical blanking. Each of $\overline{\text{CBLNK0}}$ / $\overline{\text{VBLNK0}}$ and $\overline{\text{CBLNK1}}$/$\overline{\text{VBLNK1}}$ provides one of two blanking functions, depending on the configuration of the $\overline{\text{CSYNC}}$/$\overline{\text{HBLNK}}$ pin: Composite blanking disables pixel display/capture during both horizontal and vertical retrace periods and is enabled when $\overline{\text{CSYNC}}$ is selected for composite-sync video systems. Vertical blanking disables pixel display/capture during vertical retrace periods and is enabled when $\overline{\text{HBLNK}}$ is selected for separate-sync video systems. Following reset, $\overline{\text{CBLNK0}}$ / $\overline{\text{VBLNK0}}$ and $\overline{\text{CBLNK1}}$ / $\overline{\text{VBLNK1}}$ are configured as $\overline{\text{CBLNK0}}$ and $\overline{\text{CBLNK1}}$, respectively.
$\overline{\text{CSYNC0}}$ / $\overline{\text{HBLNK0}}$, $\overline{\text{CSYNC1}}$ / $\overline{\text{HBLNK1}}$	I/O/Z	Composite sync/horizontal blanking. $\overline{\text{CSYNC0}}$ / $\overline{\text{HBLNK0}}$ and $\overline{\text{CSYNC1}}$ / $\overline{\text{HBLNK1}}$ 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, $\overline{\text{CSYNC0}}$ / $\overline{\text{HBLNK0}}$ and $\overline{\text{CSYNC1}}$ / $\overline{\text{HBLNK1}}$ are configured as high-impedance $\overline{\text{CSYNC0}}$ and $\overline{\text{CSYNC1}}$, 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.
$\overline{\text{HSYNC0}}$, $\overline{\text{HSYNC1}}$	I/O/Z	Horizontal sync. $\overline{\text{HSYNC0}}$ and $\overline{\text{HSYNC1}}$ control the video system. They can be programmed as input, output, or high impedance signals. As an input, $\overline{\text{HSYNC}}$ synchronizes the video timer to externally generated horizontal sync pulses. As an output, $\overline{\text{HSYNC}}$ 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, $\overline{\text{HSYNC0}}$ and $\overline{\text{HSYNC1}}$ are in the high-impedance state.
SCLK0, SCLK1	I	Serial data clock. SCLK0 and SCLK1 are used by the 'C80 shift register transfer (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.
$\overline{\text{VSYNC0}}$, $\overline{\text{VSYNC1}}$	I/O/Z	Vertical sync. $\overline{\text{VSYNC0}}$ and $\overline{\text{VSYNC1}}$ control the video system. They can be programmed as inputs, outputs, or high-impedance signals. As inputs, $\overline{\text{VSYNCx}}$ synchronize the frame timer to externally generated vertical-sync pulses. As outputs, $\overline{\text{VSYNCx}}$ 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, $\overline{\text{VSYNCx}}$ 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

† I = input, O = output, Z = high-impedance
‡ This pin has an internal pullup and can be left unconnected during normal operation.
§ This pin has an internal pulldown and can be left unconnected during normal operation.
¶ For proper operation, all V_{DD} and V_{SS} pins must be connected externally.

Terminal Functions (Continued)

TERMINAL NAME	TERMINAL TYPE†	DESCRIPTION
MISCELLANEOUS		
NC		No connect serves as an alignment key or is for factory use and must be left unconnected.

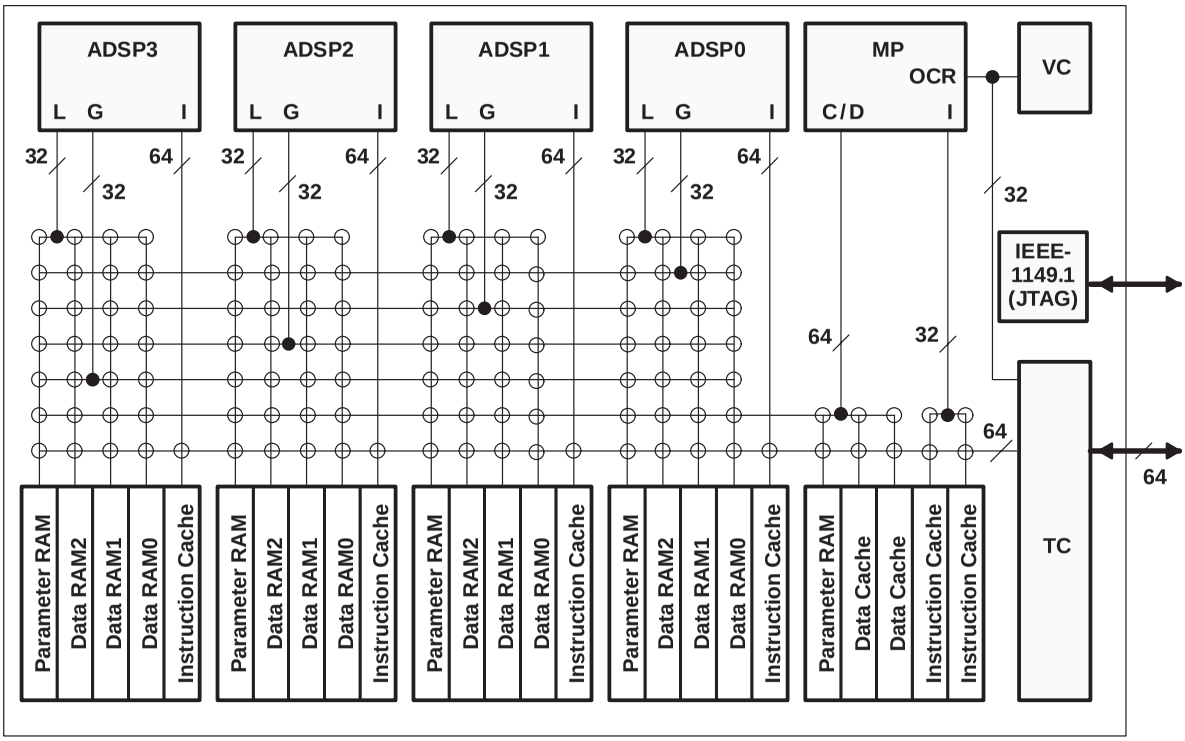
† I = input, O = output, Z = high-impedance
‡ This pin has an internal pullup and can be left unconnected during normal operation.
§ This pin has an internal pulldown and can be left unconnected during normal operation.
¶ For proper operation, all V_{DD} and V_{SS} pins must be connected externally.

architecture

Figure 1 shows the major components of the ’C80: the master processor (MP), the parallel digital signal processors (PPs), the transfer controller (TC), 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 nine 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 data path exists between the MP and the TC and VC. This allows the MP to access TC control 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.

architecture (continued)



L Local port
G Global port
I Instruction port
C/D Crossbar/data port

Figure 1. Block Diagram Showing Data Paths

architecture (continued)

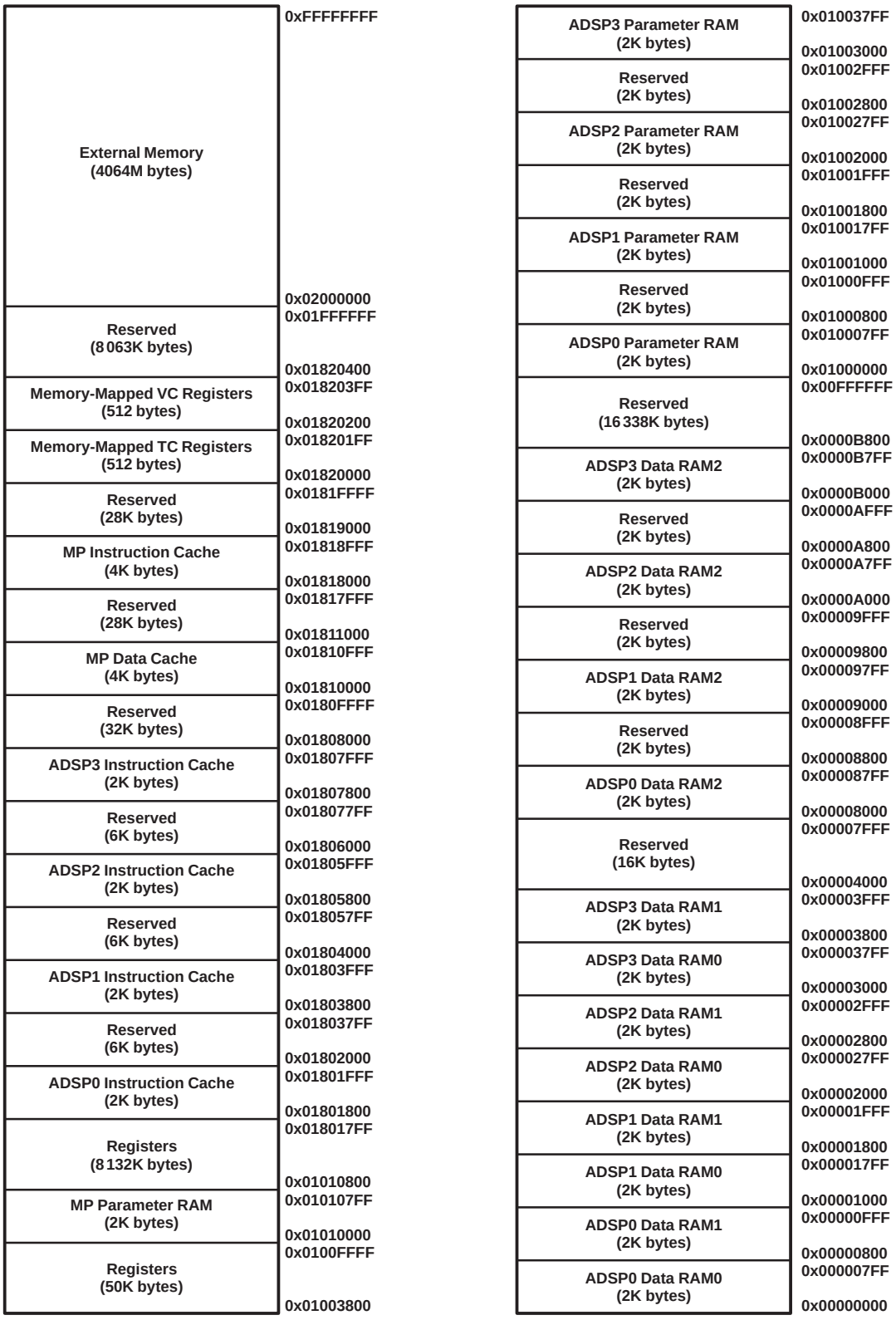


Figure 2. Memory Map

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 130 000 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
  - Least-recently-used (LRU) information replacement
  - Data writeback
- 4K-byte noncached 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
  - 50 million instructions per second (MIPS)
  - 100 million floating-point operations per second (MFLOPS)
  - Over 130 000 dhrystones/s

MP functional block diagram (continued)

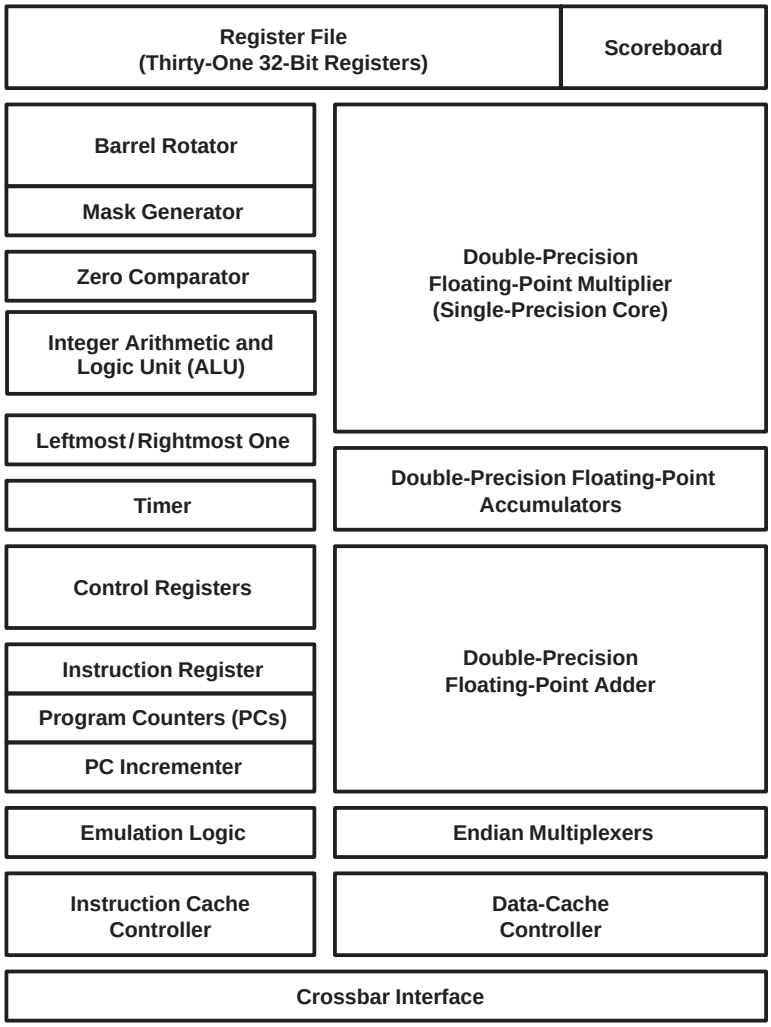


Figure 3. MP Block Diagram

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, R1 is typically 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)

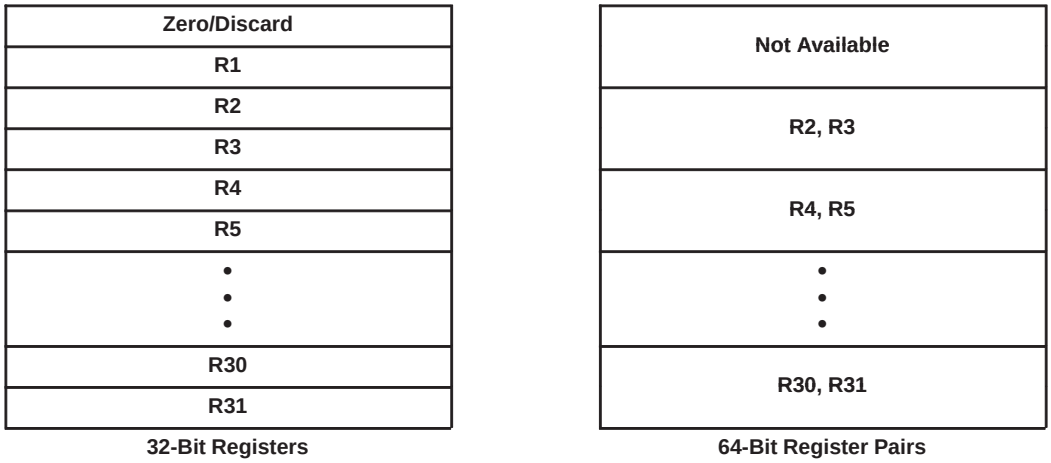
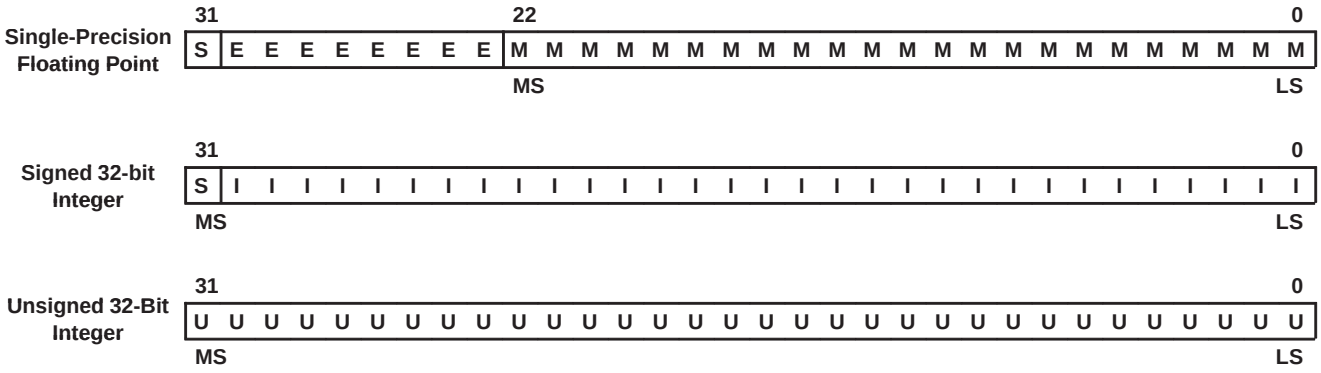


Figure 4. MP General-Purpose Registers

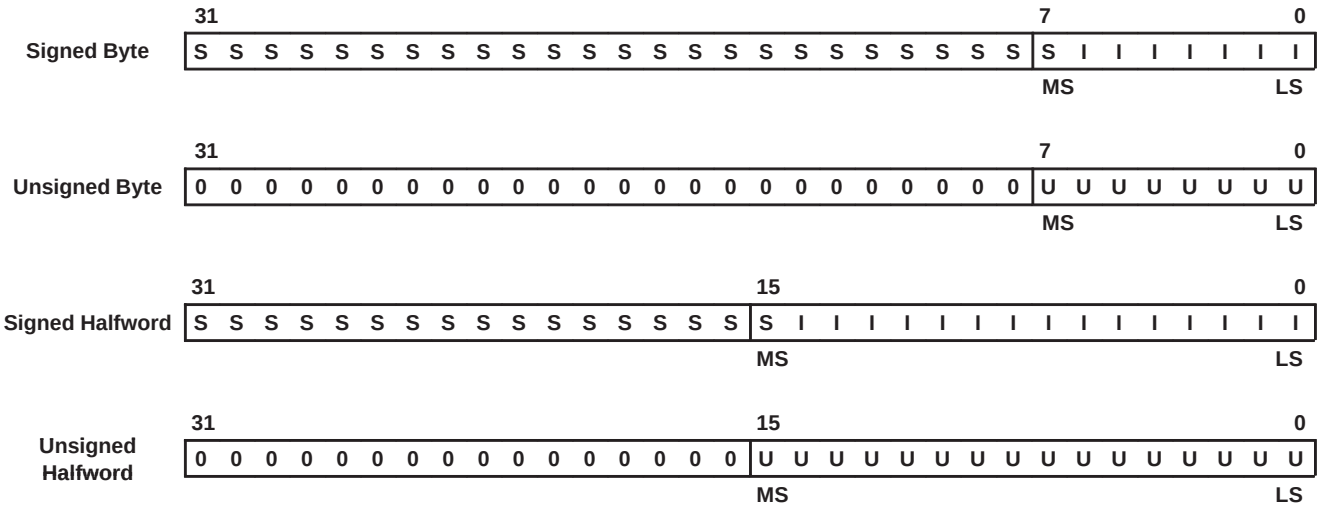
The 32-bit registers can contain signed-integer, unsigned-integer, or single-precision floating-point values. Signed and unsigned bytes and halfwords are sign-extended or zero-filled. Doublewords can be stored in a 64-bit even/odd register pair. Double-precision floating-point values are referenced using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



S	Sign bit
E	Exponent
M	Value
I	Signed integer value
U	Unsigned integer value
MS	Most significant
LS	Least signficant

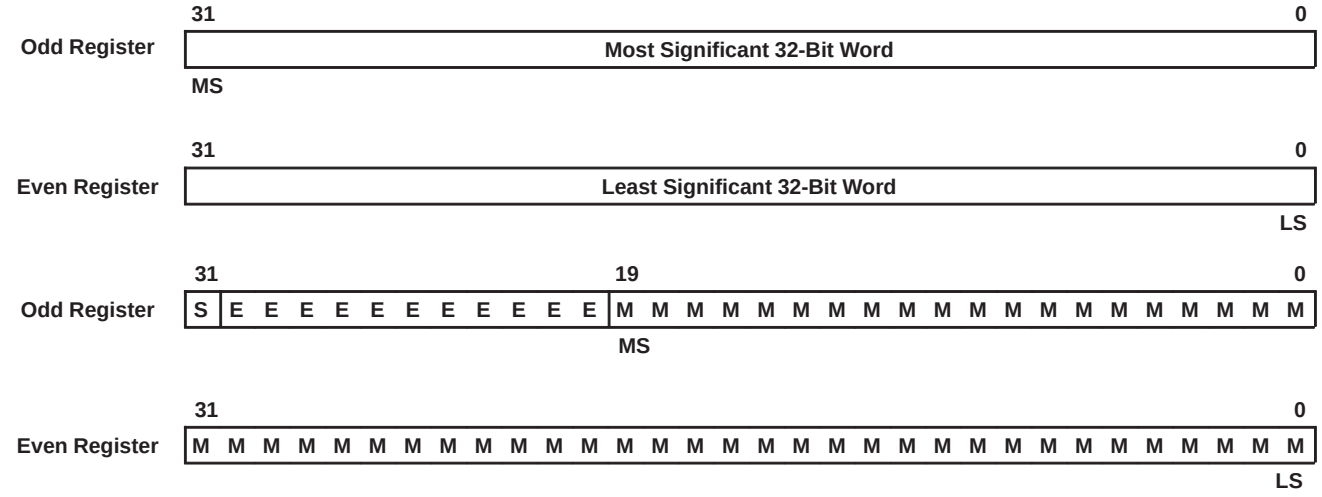
Figure 5. MP Register 32-Bit Data Formats

MP general-purpose registers (continued)



S Sign bit(s)
I Signed byte/halfword value
U Unsigned byte/halfword value
MS Most significant
LS Least signficant

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



S Sign bit(s)
E Exponent
I Signed byte/halfword value
U Unsigned byte/halfword value
MS Most significant
LS Least signficant

Figure 7. MP Register 64-Bit Data Formats

MP double-precision floating-point accumulators

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

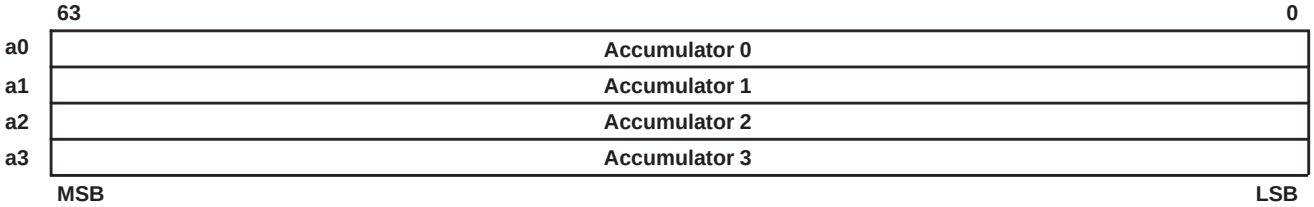


Figure 8. Double-Precision Floating-Point Accumulators

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

NUMBER	NAME	DESCRIPTION	NUMBER	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 Register	0x0030	MPC	Emulator Exception Program Counter
0x0005	—	Reserved	0x0031	MIP	Emulator Exception Instruction Pointer
0x0006	IE	Interrupt Enable Register	0x0032	—	Reserved
0x0007	—	Reserved	0x0033	ECOMCNTL	Emulator Communication Control
0x0008	FPST	Floating-Point Status	0x0034	ANASTAT	Emulation Analysis Status Register
0x0009	—	Reserved	0x0035–0x0038	—	Reserved
0x000A	PPERROR	PP Error Register	0x0039	BRK1	Emulation Breakpoint 1 Register
0x000B	—	Reserved	0x003A	BRK2	Emulation Breakpoint 2 Register
0x000C	—	Reserved	0x003B–0x01FF	—	Reserved
0x000D	PKTREQ	Packet-Transfer Request Register	0x0200 – 0x020F	iCACHET	Instruction Cache Tags 0 to 15
0x000E	TCOUNT	Current Counter Value	0x0300	iCACHEL	Instruction Cache LRU Register
0x000F	TSCALE	Counter Reload Value	0x0400–0x040F	dCACHET	Data Cache Tags 0 to 15
0x0010	FLTOP	Faulting Operation	0x0500	dCACHEL	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 Data (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 packet transfer (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 SMJ320C80 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	p2	PP2 message interrupt	f1	Frame-timer 1 interrupt
x4	External interrupt 4 ($\overline{\text{LINT4}}$)	p1	PP1 message interrupt	f0	Frame-timer 0 interrupt
x3	External interrupt 3 ($\overline{\text{EINT3}}$)	p0	PP0 message interrupt	fx	Floating-point inexact
bp	Bad packet transfer	io	Integer overflow	fu	Floating-point underflow
pb	Packet transfer busy	mf	Memory fault	fo	Floating-point overflow
pc	Packet transfer complete	x2	External interrupt 2 ($\overline{\text{EINT2}}$)	fz	Floating-point divide-by-zero
mi	MP message interrupt	x1	External interrupt 1 ($\overline{\text{EINT1}}$)	fi	Floating-point invalid
p3	PP3 message interrupt	ti	MP timer interrupt	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		

pe	PP error	p2	PP2 message interrupt	f1	Frame-timer 1 interrupt
x4	External interrupt 4 ($\overline{\text{LINT4}}$)	p1	PP1 message interrupt	f0	Frame-timer 0 interrupt
x3	External interrupt 3 ($\overline{\text{EINT3}}$)	p0	PP0 message interrupt	fx	Floating-point inexact
bp	Bad packet transfer	io	Integer overflow	fu	Floating-point underflow
pb	Packet transfer busy	mf	Memory fault	fo	Floating-point overflow
pc	Packet transfer complete	x2	External interrupt 2 ($\overline{\text{EINT2}}$)	fz	Floating-point divide-by-zero
mi	MP message interrupt	x1	External interrupt 1 ($\overline{\text{EINT1}}$)	fi	Floating-point invalid
p3	PP3 message interrupt	ti	MP timer interrupt	ie	Global-interrupt enable

Figure 12. INTPEN Register

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

FPST contains status and control information for the floating-point unit (FPU) as shown in Figure 13. Bits 17–21 are read/write 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	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
dest					ai	az	ao	au	ax	sm	fs	vm	drm		opcode					e1	e0	pd		rm			mo	i	z	o	u	x

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

Figure 13. FPST Register

PP error (PPERROR) register (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	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
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	PPhalted
I	PP illegal instruction
f	PP fault type
0	icache
1	Direct external access (DEA)

Figure 14. PPERROR Register

packet-transfer request (PKTREQ) register (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.

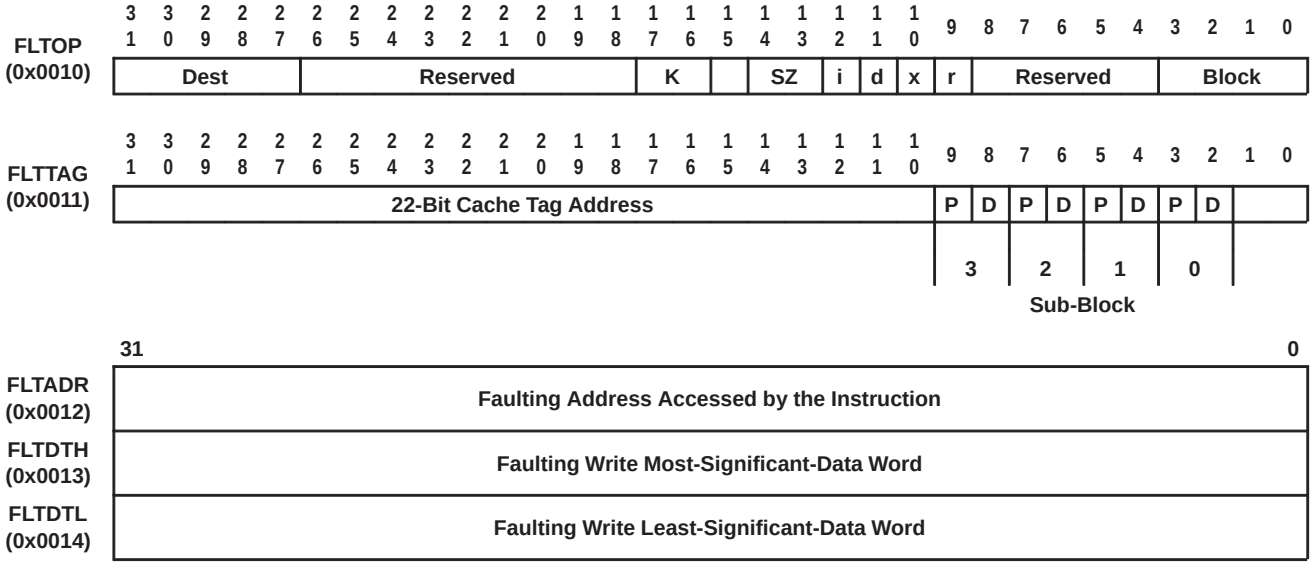
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											I	F	S	Q	P

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.

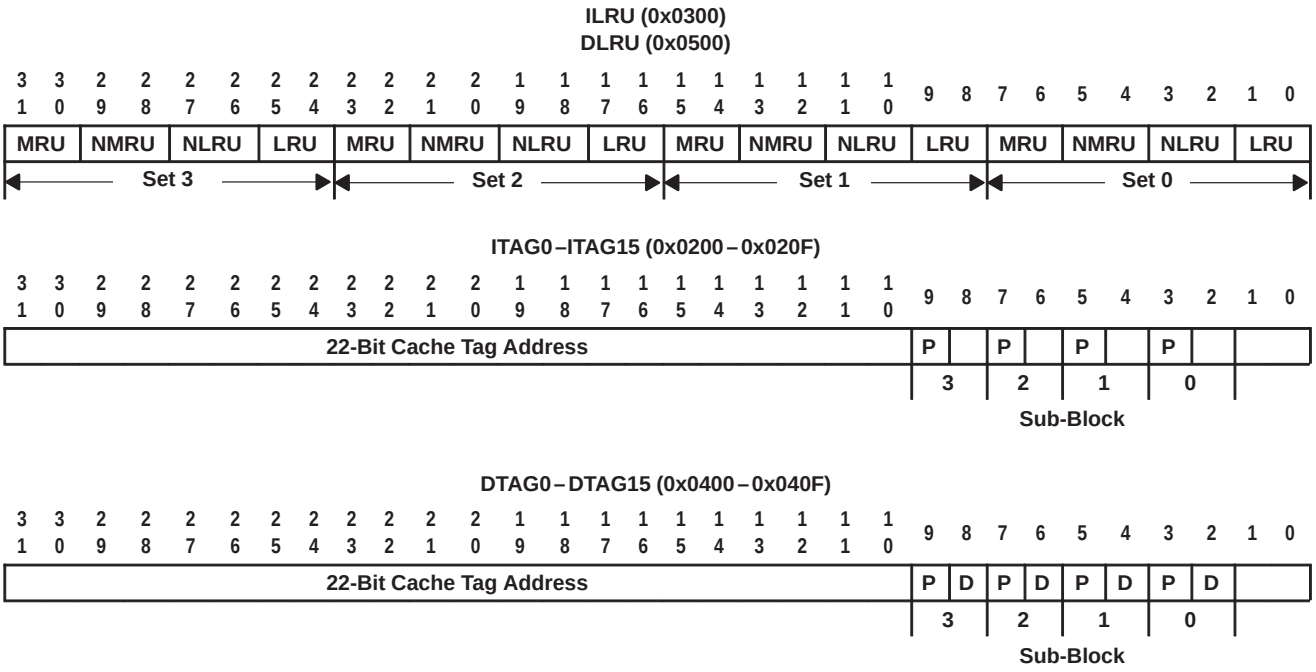


Dest	Destination Register Number
K	Kind of Operation:
	00 – load
	01 – unsigned load
	10 – store
	11 – cache flush/clean
SZ	Size of Data:
	00 – 8-bit
	01 – 16-bit
	10 – 32-bit
	11 – 64-bit
i	MP icache fault
d	MP dcache fault
x	DEA Fault
r	Modified return sequence
Block	Faulting block number
P	Sub-block 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.



MRU	**Most-recently-used**	**LRU**	**Least-recently-used**
NMRU	**Next most-recently-used**	**P**	**Sub-block present**
NLRU	**Next least-recently-used**	**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.

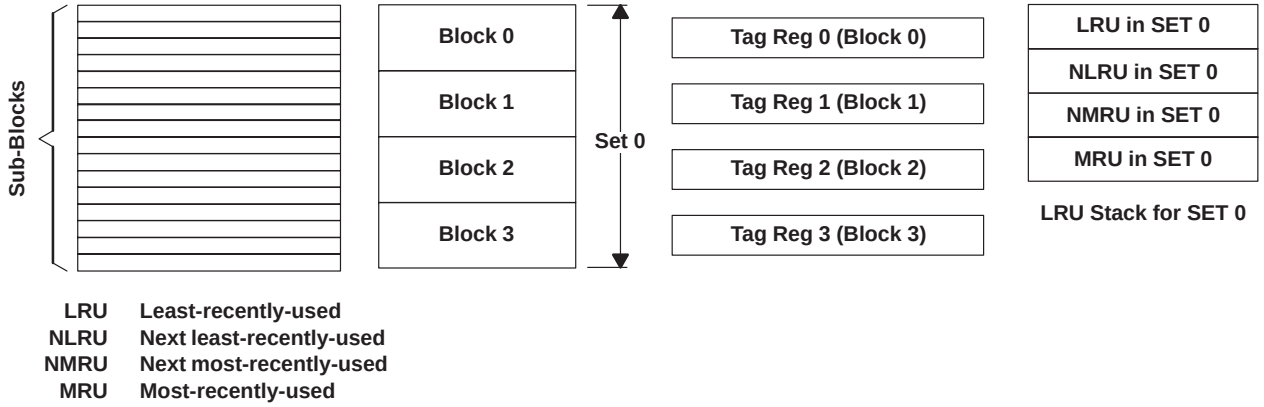


Figure 18. MP Cache Architecture (x4 Sets)

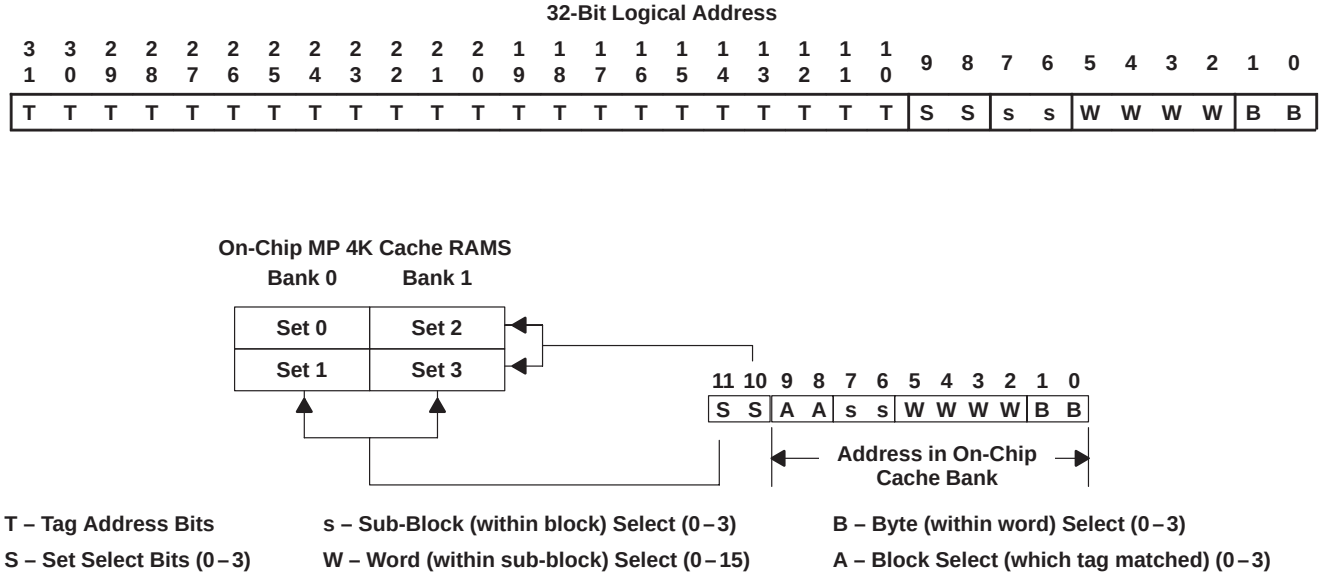


Figure 19. MP Cache Addressing

MP parameter RAM

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

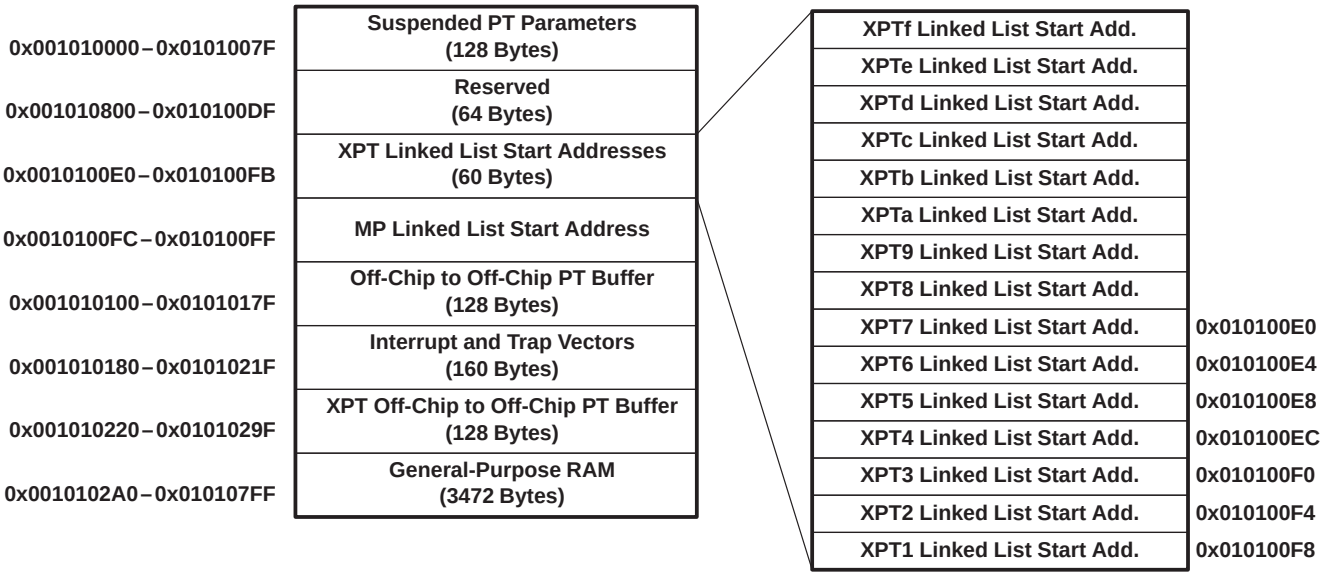


Figure 20. MP Parameter RAM

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	Reserved
9	f1	0x010101A4	Reserved
10	ti	0x010101A8	MP timer interrupt
11	x1	0x010101AC	External interrupt 1 ($\overline{\text{EINT1}}$)
12	x2	0x010101B0	External interrupt 2 ($\overline{\text{EINT2}}$)
14	mf	0x010101B8	Memory fault
15	io	0x010101BC	Integer overflow
16	p0	0x010101C0	PP0 message interrupt
17	p1	0x010101C4	PP1 message interrupt
18	p2	0x010101C8	Reserved
19	p3	0x010101CC	Reserved
25	mi	0x010101E4	MP message interrupt
26	pc	0x010101E8	Packet-transfer complete
27	pb	0x010101EC	Packet-transfer busy
28	bp	0x010101F0	Bad packet transfer
29	x3	0x010101F4	External interrupt 3 ($\overline{\text{EINT3}}$)
30	x4	0x010101F8	External interrupt 4 ($\overline{\text{LINT4}}$)
31	pe	0x010101FC	PP error

Table 3. Nonmaskable Traps

TRAP NUMBER	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.

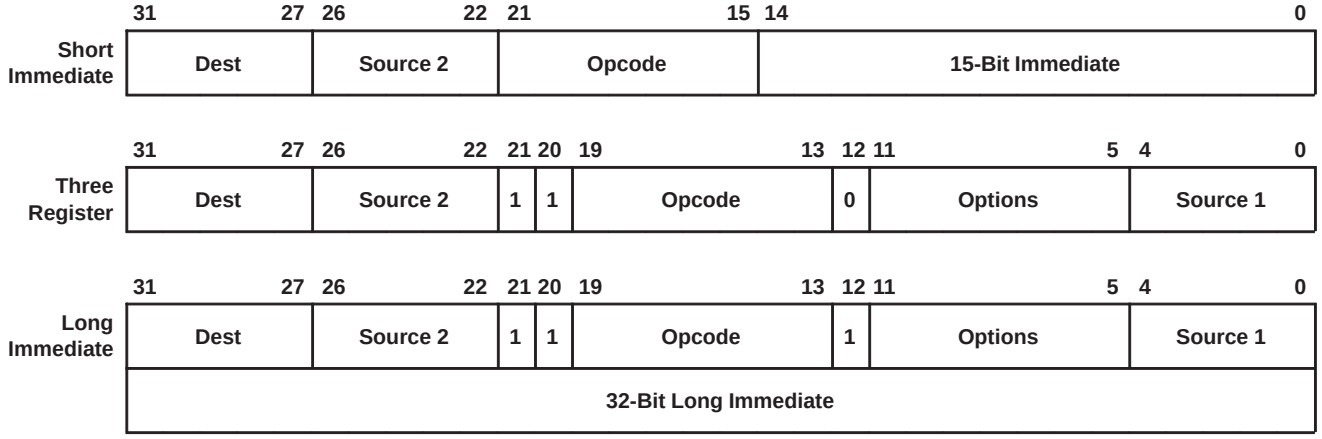


Figure 21. MP Opcode Formats

MP opcode summary

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

MP opcode summary (continued)

Table 4. Short-Immediate Opcodes

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
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	1	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	09	08	07	06	05	04	03	02	01	00
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	I	i	n	Endmask					Rotate				
shift.dm	Dest					Source					1	1	0	0	0	1	0	0	1	I	i	n	Endmask					Rotate				
shift.ds	Dest					Source					1	1	0	0	0	1	0	1	0	I	i	n	Endmask					Rotate				
shift.ez	Dest					Source					1	1	0	0	0	1	0	1	1	I	i	n	Endmask					Rotate				
shift.em	Dest					Source					1	1	0	0	0	1	1	0	0	I	i	n	Endmask					Rotate				
shift.es	Dest					Source					1	1	0	0	0	1	1	0	1	I	i	n	Endmask					Rotate				
shift.iz	Dest					Source					1	1	0	0	0	1	1	1	0	I	i	n	Endmask					Rotate				
shift.im	Dest					Source					1	1	0	0	0	1	1	1	1	I	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	–	–	–	–	–	Source				
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	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
vadd	Mem Src/Dst					Source2/Dest					1	1	1	1	0	–	0	0	0	I	–	m	P	–	d	m	s	Source1					
vsub	Mem Src/Dst					Source2/Dest					1	1	1	1	0	–	0	0	1	I	–	m	P	–	d	m	s	Source1					
vmpy	Mem Src/Dst					Source2/Dest					1	1	1	1	0	–	0	1	0	I	–	m	P	–	d	m	s	Source1					
vmsub	Mem Src/Dst					Dest					1	1	1	1	0	a	0	1	1	I	a	m	P	Z	–	m	–	Source1					
vrnd(FP)	Mem Src/Dst					Dest					1	1	1	1	0	a	1	0	0	I	a	m	P	PD		m	s	Source1					
vrnd(Int)	Mem Src/Dst					Dest					1	1	1	1	0	–	1	0	1	I	–	m	P	–	d	m	s	Source1					
vmac	Mem Src/Dst					Source2					1	1	1	1	0	a	1	1	0	I	a	m	P	Z	–	m	–	Source1					
vmac	Mem Src/Dst					Source2					1	1	1	1	0	a	1	1	1	I	a	m	P	Z	–	m	–	Source1					
fadd	Dest					Source2					1	1	1	1	1	0	0	0	0	I	–	PD		P2		P1		Source1					
fsub	Dest					Source2					1	1	1	1	1	0	0	0	1	I	–	PD		P2		P1		Source1					
fmpy	Dest					Source2					1	1	1	1	1	0	0	1	0	I	–	PD		P2		P1		Source1					
fdiv	Dest					Source2					1	1	1	1	1	0	0	1	1	I	–	PD		P2		P1		Source1					
frndx	Dest					–	–	–	–	–	1	1	1	1	1	0	1	0	0	I	–	PD		RM		P1		Source1					
fcmp	Dest					Source2					1	1	1	1	1	0	1	0	1	I	–	–		P2		P1		Source1					
fsqrt	Dest					–	–	–	–	–	1	1	1	1	1	0	1	1	1	I	–	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)
I	Long immediate 32-bit data
m	Parallel memory operation specifier
Mem Src/Dst	Vector store or load source/dst register
Dest	Destination register
P	Destination 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
  - Discrete Cosine Transform (DCT)
  - Fast Fourier Transform (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
  - 16 x 16 integer multiplier (optional dual 8 x 8)
  - 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 and 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)

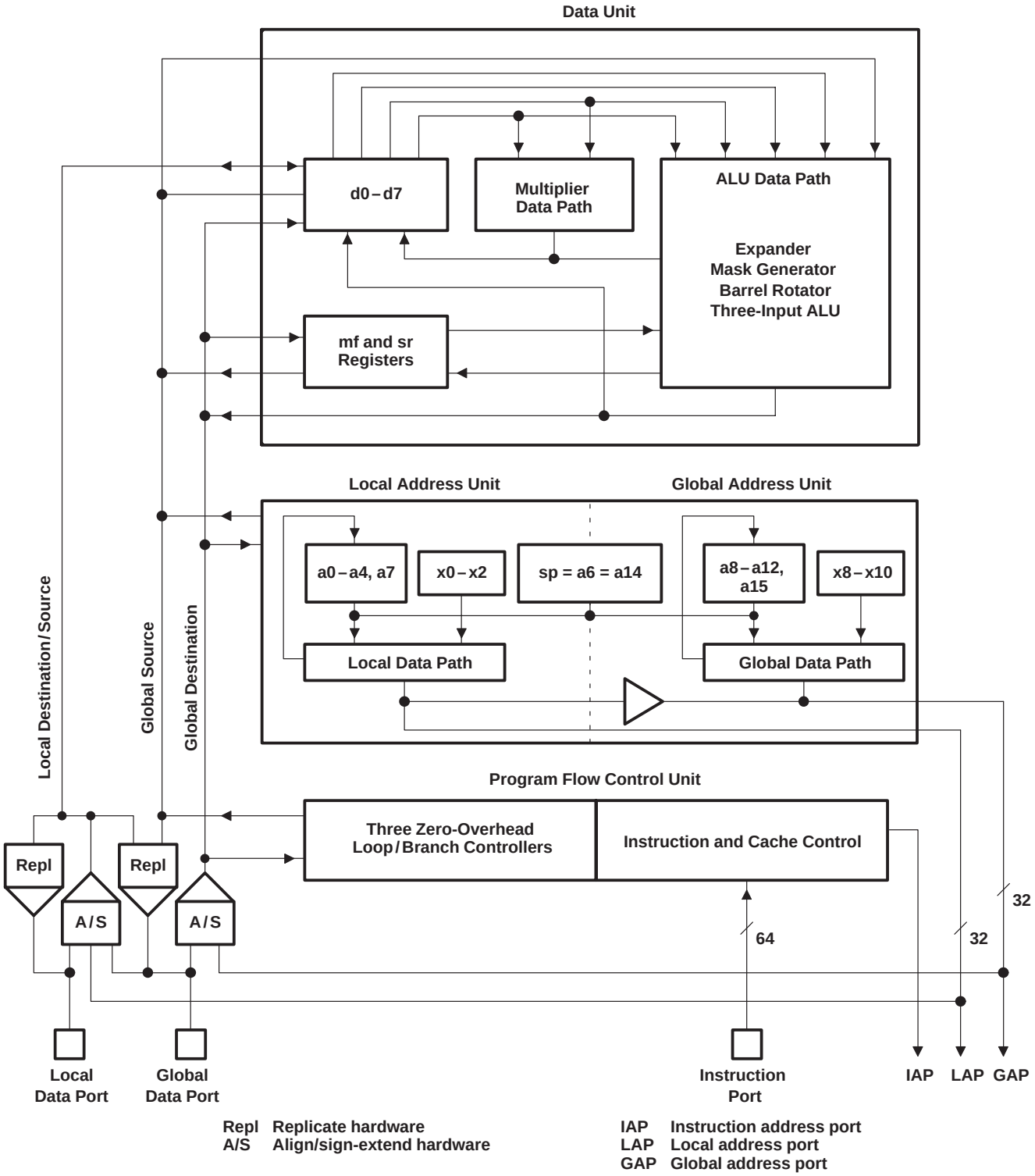


Figure 22. PP Block Diagram

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.

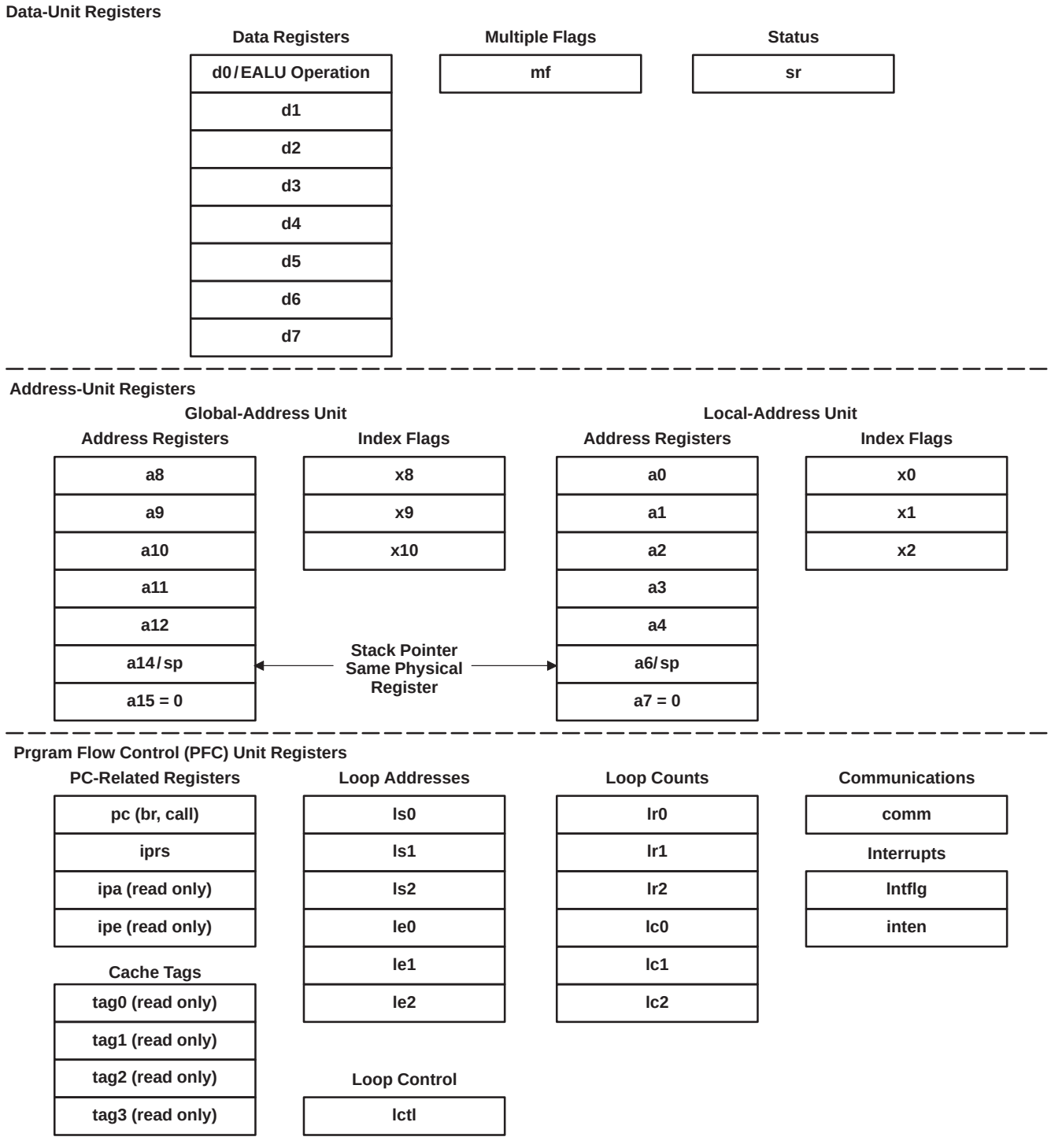


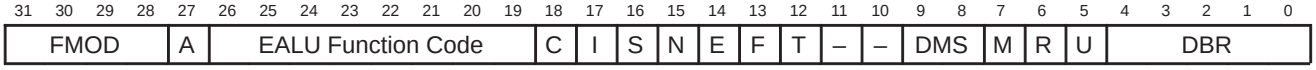
Figure 23. PP Registers

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 extended ALU (EALU) operations.

d0 register

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



FMOD Function modifiers
A Arithmetic enable
C EALU carry-In
I Invert-carry-In
S Sign extend
N Nonmultiple mask
E Explicit multiple carry-in
F Expanded multiple flags
DMS Default multiply shift amount
M Split multiply
R Rounded multiply
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 can be expanded to generate a mask for the ALU. Figure 25 shows the mf register format.

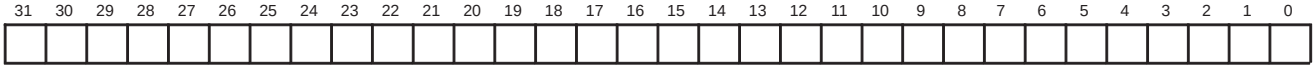
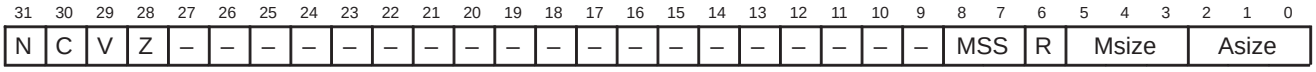


Figure 25. mf Register Format

status register (sr)

The sr contains status and control bits for the PP ALU. See Figure 26.



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

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–x3 are used by the local-address unit and registers x8–x9 are used by the global-address unit.

stack pointer (sp)

The sp contains the address of the top 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.

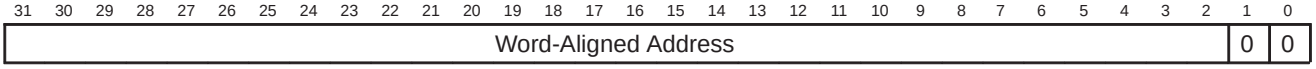


Figure 27. sp Register Format

zero registers

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.

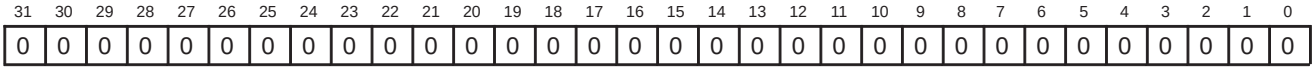
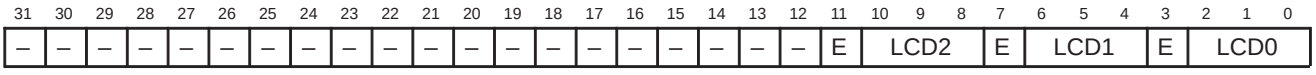


Figure 28. Zero Registers

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.



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.

pipeline registers (continued)

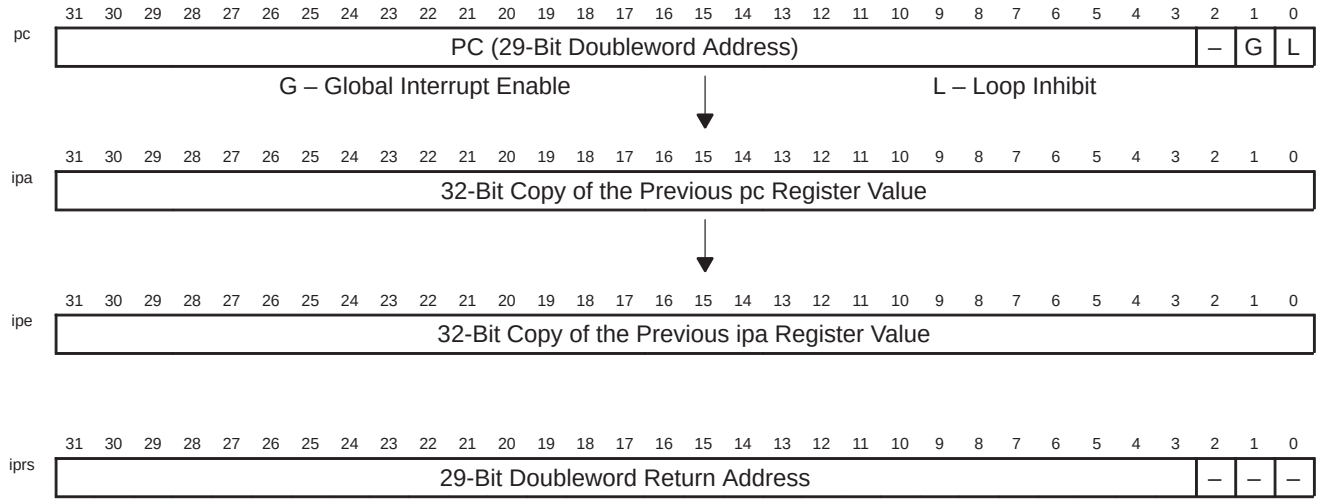
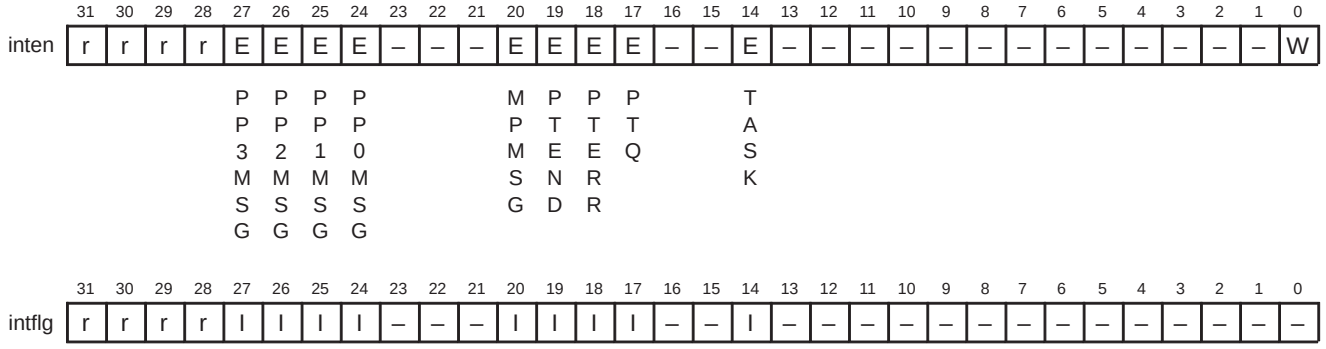


Figure 30. Pipeline Registers

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.



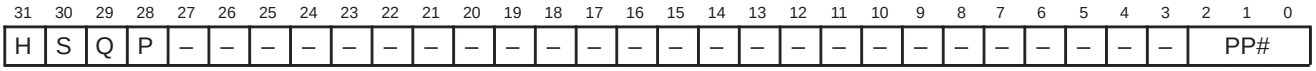
r **Reserved (write as 0)**
E **Enable interrupt**
W **Write mode**
0 – writing 1 clears intflg
1 – writing 1 sets intflg
PPnMSG **PPn message interrupt**

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

communication (comm) register

The comm register contains the packet-transfer handshake bits and PP indicator bits.



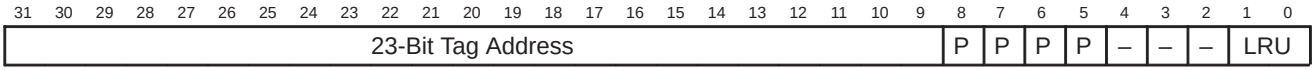
H **High-priority packet transfer**
S **Packet-transfer suspend**
Q **Packet transfer queued**
P **Submit packet transfer request**

PP# **PP Number (read only)**
000 – PP0 **010 – PP2**
001 – PP1 **011 – PP3**
1xx – Not implemented

Figure 32. comm Register

cache-tag registers

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



P **Present bit**
LRU **Least-recently-used code**
00 – Most-recently-used (MRU)
10 – next LRU
01 – next MRU (NMRU)
11 – LRU

Sub-Block # **3** **2** **1** **0**

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.

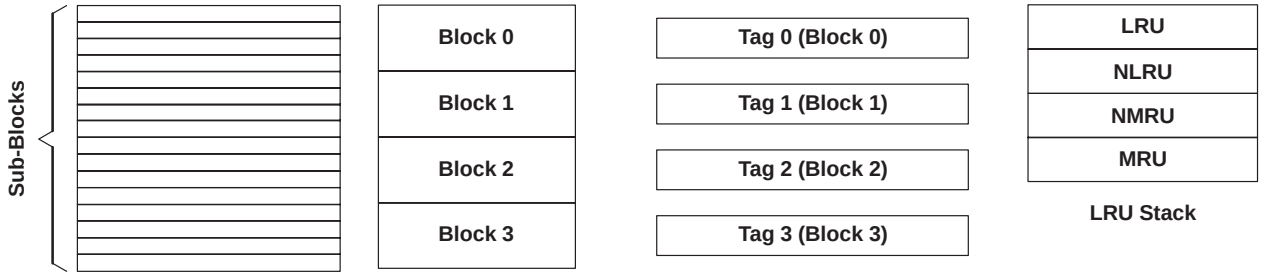
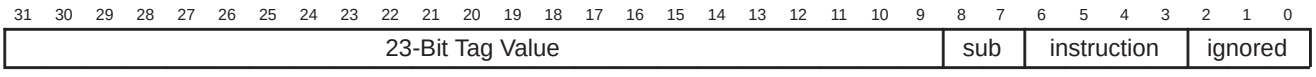


Figure 34. PP Cache Architecture



sub – sub-block

Figure 35. PP 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 35 shows the parameter RAM address map.

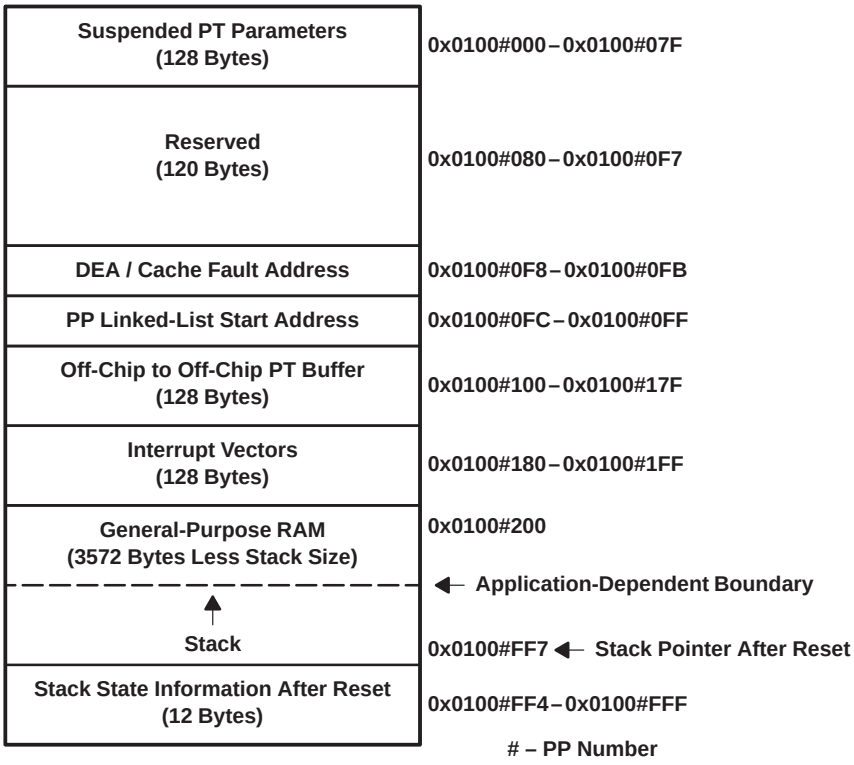


Figure 36. PP Parameter RAM Address Map

PP-interrupt vectors

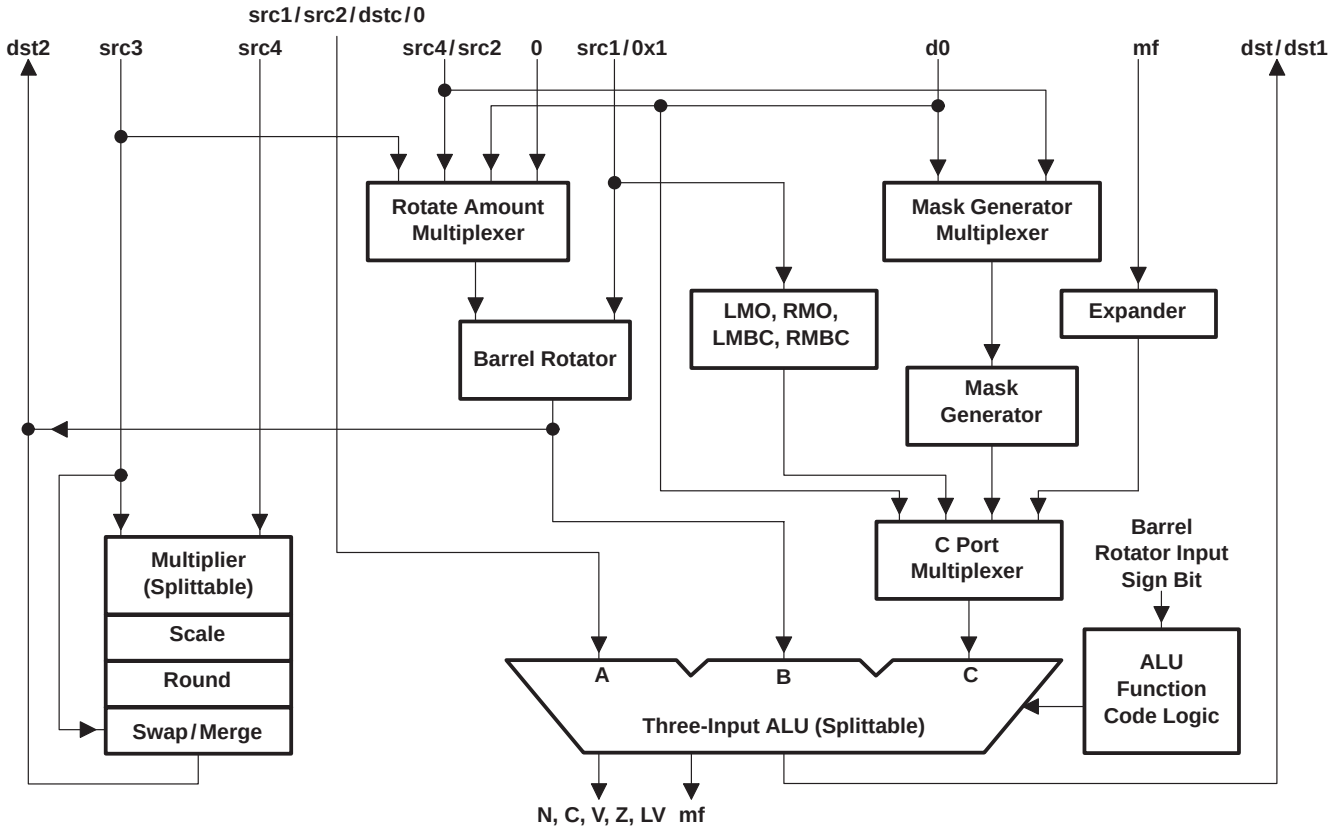
The PP interrupts and their vector addresses are shown in Table 8.

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	0x0101#1E4	PP1 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, multiple flag (mf) expander, left/rightmost one and left/rightmost bit-change logic, and several multiplexers. Figure 37 shows the data-unit block diagram.



src1	Any register, D reg only for left/right most one (LMO/RMO), left/right most bit change (LMBC/RMBC) hardware		
scr2	D reg or sometimes 5/32-bit immediate	dst2	D reg only
scr3	D reg only	dstc	D reg only (destination companion reg source)
scr4	D reg only	0x1	Constant
dst/dst1	Any register	0	Constant
		d0	5 LSBs of d0

Figure 37. Data-Unit Block Diagram

PP data-unit architecture (continued)

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.

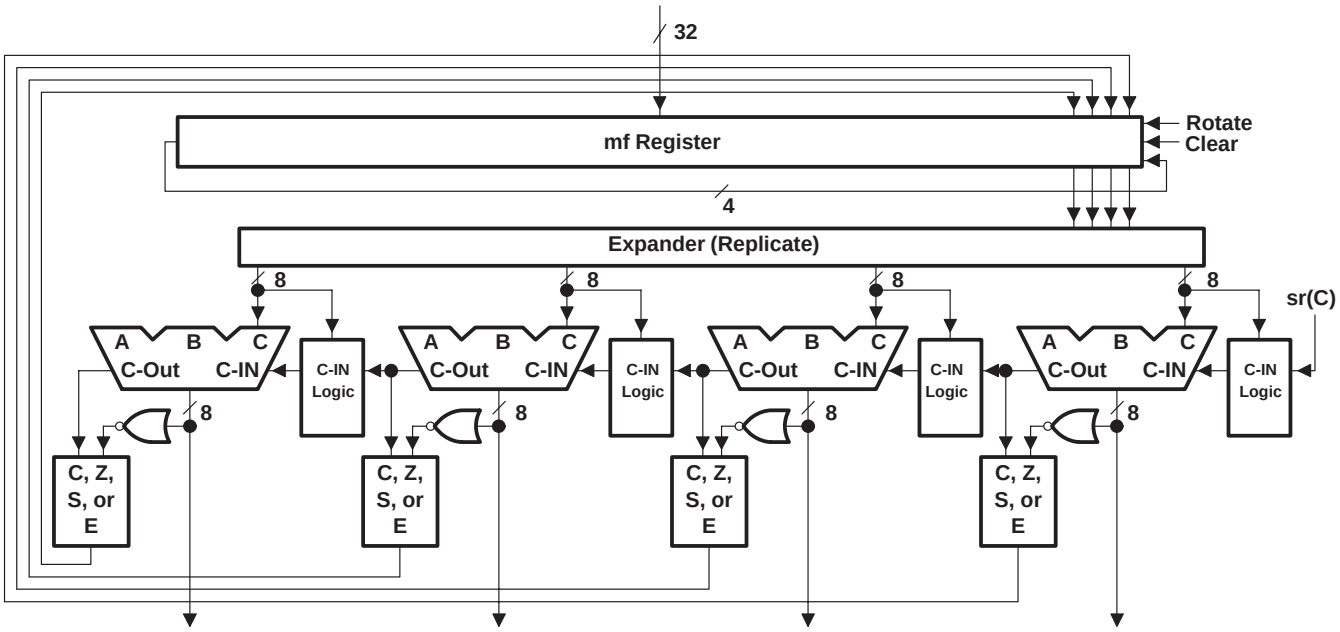


Figure 38. Multiple-Byte Arithmetic Data Flow

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.

Figure 39. 16 x 16 Multiplier Data Formats

PP multiplier (continued)

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.

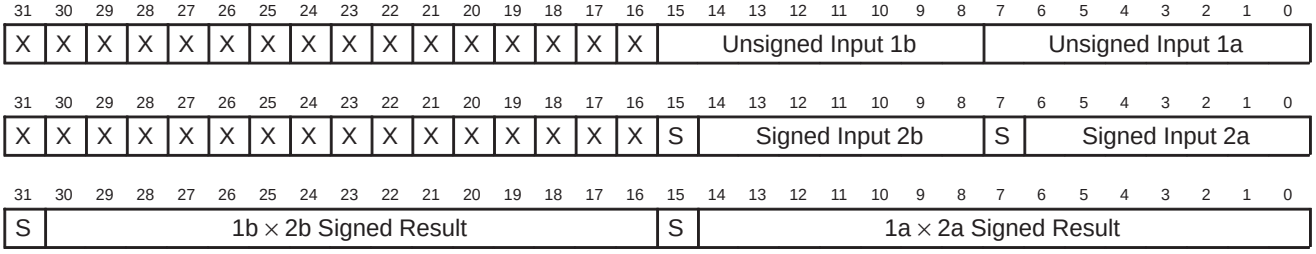


Figure 40. Signed Split Multiply Data Formats

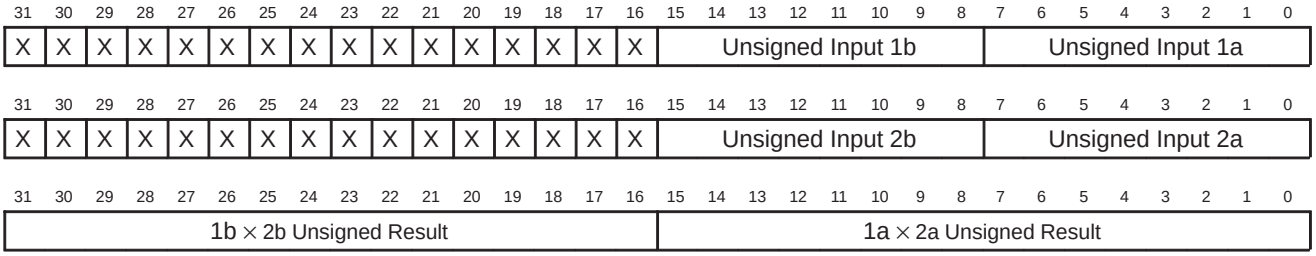


Figure 41. Unsigned Split Multiply Data Formats

PP program-flow-control unit architecture

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.

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).

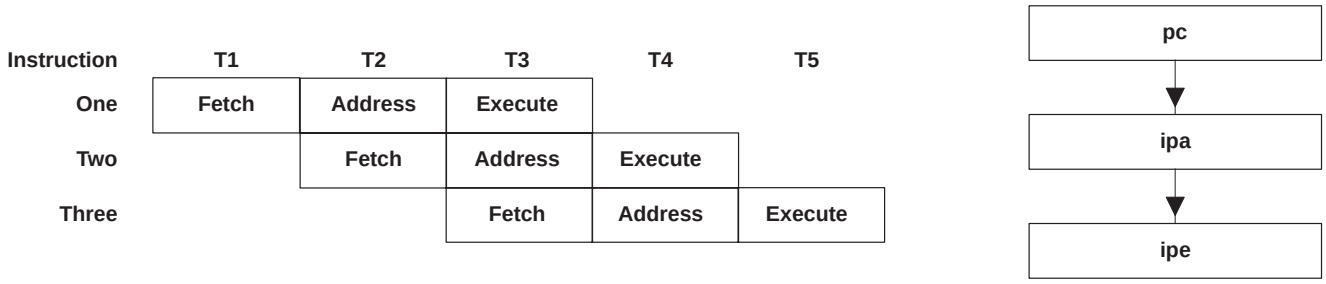


Figure 42. FAE-Instruction Pipeline

PP program-flow-control unit architecture (continued)

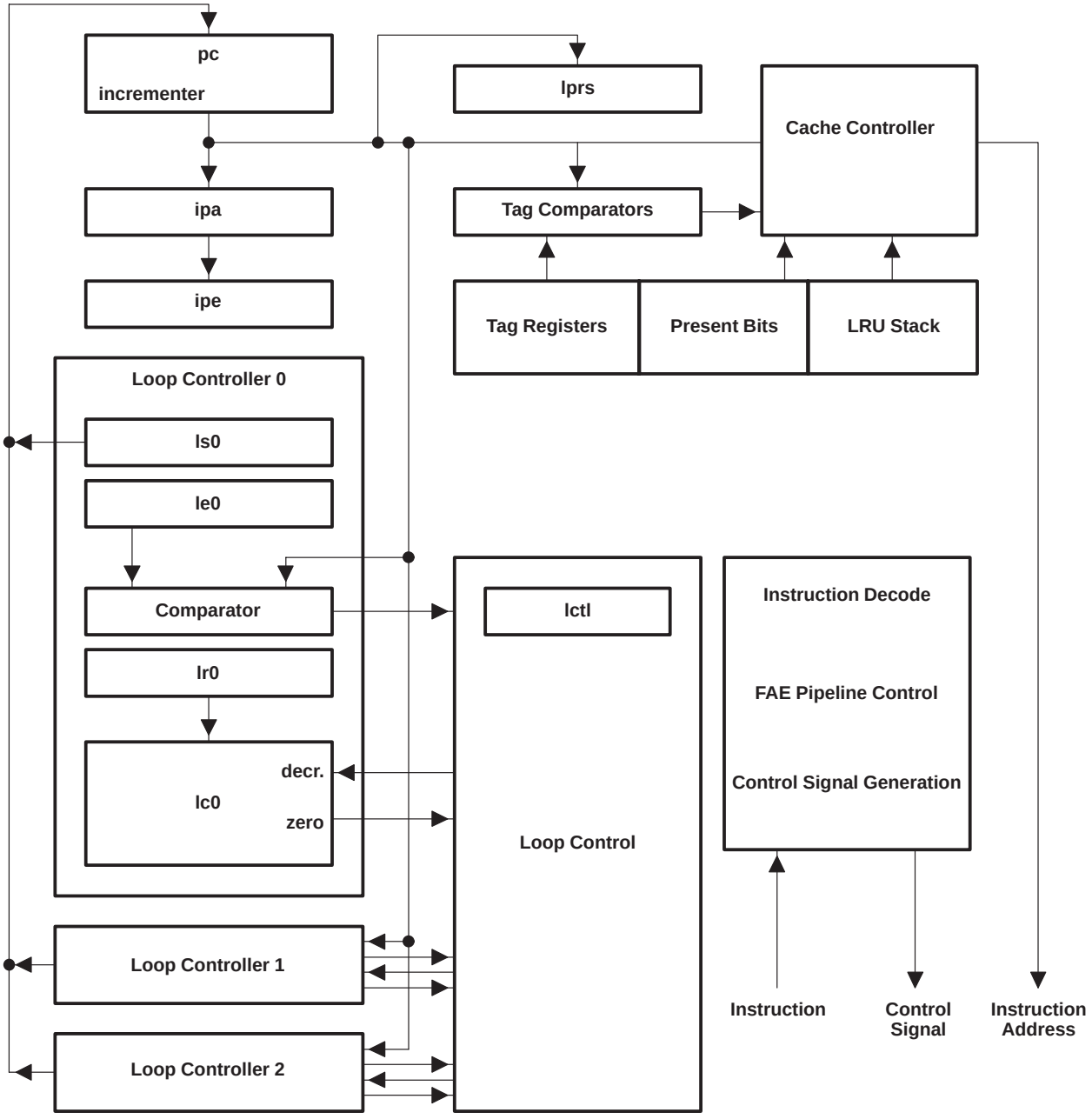


Figure 43. Program-Flow-Control Unit Block Diagram

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.

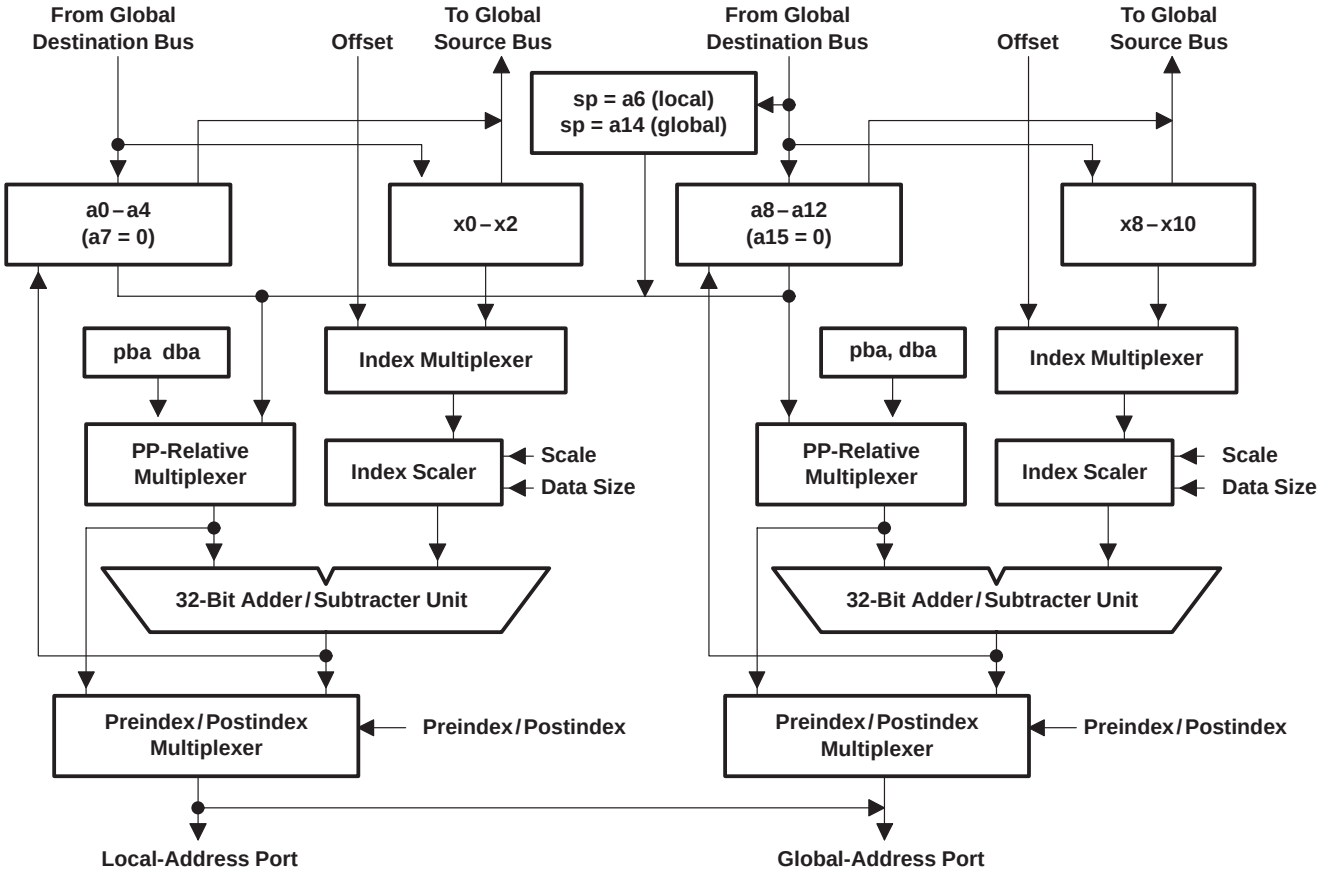


Figure 44. Address-Unit Architecture

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.

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

Operation	Base set ALUs
Description	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 16 arithmetic operations combined with one of 16 function modifiers.
Syntax	dst = [fmod] [[[cond [.pro]]]] ALU_EXPRESSION
Examples	`d6 = (d6 ^ d4) & d2` `d3 = [nn.nv] d1 –1`
Operation	EALU || ROTATE
Description	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.
Syntax	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])
Examples	`d7 = [nn] ealu(d2, d6 = [nn] d3\\d1, %d4)` `d3 = mzc ealu(mylabel: d4 + (d5\\d6 & %d7) || d1 = d5\\d6)`
Operation	MPY || ADD
Description	Perform a 16x16 multiply with optional parallel add or subtract. Condition code applies to both multiply and add.
Syntax	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]]
Example	`d7 = u d6 * d5 || d5 = d4 – d1`
Operation	MPY || SADD
Description	Perform a 16x16 multiply with a parallel right-shift and add or subtract. Condition code applies to multiply, shift, and add.
Syntax	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
Examples	`d7 = u d6 * d5 || d5 = d4 – d1 >> –d0`
Operation	MPY || EALU
Description	Perform a multiply and an optional parallel EALU. Multiply can use rounding, scaling, or splitting features.
Syntax	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)
Examples	`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`
Operation	divi
Description	Perform one iteration of unsigned divide algorithm. Generates one quotient bit per execution using iterative subtraction.
Syntax	dst1 = [[[cond [.pro]]]] divi (src2, dst2 = [[cond]] src1 [[[n]] src1 –1])
Examples	`d3 = divi (d1, d2 = d2)` `d3 = divi (d1, d2 = d3[n]d2)`
Misc. Operations	dint; eint; nop
Description	Globally disable interrupts; globally enable interrupts; do nothing in the data unit

Syntax	`dint`
	`eint`
	`nop`

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 complement 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 least significant byte or least significant halfword 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

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	37–0	
0	1	1	oper				src3				dst2				dst1				src1				src4			src2	Parallel Transfers	A. Six-Operand (MPYIIADD, etc.)
1	class			A	ALU Operation										dst				src1		0	imm. src2				Parallel Transfers	B. Base Set ALU (5-Bit Immediate)
1	class			A	ALU Operation										dst				src1		1	0	–	src2		Parallel Transfers	C. Base Set ALU (Register src2)
1	class			A	ALU Operation										dst				src1		1	1	dstbank		s1bnk	cond	32-Bit Immediate	D. Base Set ALU (32-Bit Immediate)
1	0	0	0	1	–	0	–	0	–	0	–	0	–	–	–	–	–	–	0	Operation						Parallel Transfers	E. Miscellaneous
0	0	Reserved																									
0	1	0	Reserved																								

Transfer Formats

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

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 eight 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 s1bnk 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)
Adstbank	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 items 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 and indicates 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 least significant bits (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 most significant bits (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, Adstbank, 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)					
CODING		REGISTER	CODING		REGISTER
BANK	REG		BANK	REG	
0000	000	a0	0100	000	d0
0000	001	a1	0100	001	d1
0000	010	a2	0100	010	d2
0000	011	a3	0100	011	d3
0000	100	a4	0100	100	d4
0000	101	reserved	0100	101	d5
0000	110	a6 (sp)	0100	110	d6
0000	111	a7 (zero)	0100	111	d7
0001	000	a8	0101	000	reserved
0001	001	a9	0101	001	sr
0001	010	a10	0101	010	mf
0001	011	a11	0101	011	reserved
0001	100	a12	0101	100	reserved
0001	101	reserved	0101	101	reserved
0001	110	a14 (sp)	0101	110	reserved
0001	111	a15 (zero)	0101	111	reserved
0010	000	x0	0110	000	reserved
0010	001	x1	0110	001	reserved
0010	010	x2	0110	010	reserved
0010	011	reserved	0110	011	reserved
0010	100	reserved	0110	100	reserved
0010	101	reserved	0110	101	reserved
0010	110	reserved	0110	110	reserved
0010	111	reserved	0110	111	reserved
0011	000	x8	0111	000	pc/call
0011	001	x9	0111	001	ipa/br
0011	010	x10	0111	010	ipe †
0011	011	reserved	0111	011	iprs
0011	100	reserved	0111	100	inten
0011	101	reserved	0111	101	intflg
0011	110	reserved	0111	110	comm
0011	111	reserved	0111	111	lctl

UPPER REGISTERS (MSB OF BANK = 1)					
CODING		REGISTER	CODING		REGISTER
BANK	REG		BANK	REG	
1000	000	reserved	1100	000	lc0
1000	001	reserved	1100	001	lc1
1000	010	reserved	1100	010	lc2
1000	011	reserved	1100	011	reserved
1000	100	reserved	1100	100	lr0
1000	101	reserved	1100	101	lr1
1000	110	reserved	1100	110	lr2
1000	111	reserved	1100	111	reserved
1001	000	reserved	1101	000	lrse0
1001	001	reserved	1101	001	lrse1
1001	010	reserved	1101	010	lrse2
1001	011	reserved	1101	011	reserved
1001	100	reserved	1101	100	lrs0
1001	101	reserved	1101	101	lrs1
1001	110	reserved	1101	110	lrs2
1001	111	reserved	1101	111	reserved
1010	000	reserved	1110	000	ls0
1010	001	reserved	1110	001	ls1
1010	010	reserved	1110	010	ls2
1010	011	reserved	1110	011	reserved
1010	100	reserved	1110	100	le0
1010	101	reserved	1110	101	le1
1010	110	reserved	1110	110	le2
1010	111	reserved	1110	111	reserved
1011	000	reserved	1111	000	reserved
1011	001	reserved	1111	001	reserved
1011	010	reserved	1111	010	reserved
1011	011	reserved	1111	011	reserved
1011	100	reserved	1111	100	tag0 †
1011	101	reserved	1111	101	tag1 †
1011	110	reserved	1111	110	tag2 †
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. See Figure 45.

Table 16. Six-Operand Format Operation Codes

oper FIELD BIT				OPERATION TYPE
60	59	58	57	
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	SPY || 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. See Figure 45.

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 register
- dst Destination D register or any register if dstbank or Adstbank is used with destination.
- src2 Source D register or immediate
- srd1 Source D register or any register if As1bank is used or any lower register 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. See Table 9 for a list of PP operators and Figure 45 for PP opcode formats.

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				MODIFICATION PERFORMED
58	56	54	52	
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)	%!	Modified mask generator
lmbc	Leftmost-bit change	rmbc	Rightmost-bit change
lmo	Leftmost one	rmo	Rightmost one

miscellaneous operation code

For data-unit opcode format E, the operation field selects one of the miscellaneous operations.

Table 21. Miscellaneous Operation Codes

OPCODE BITS					MNEMONIC	OPERATION
43	42	41	40	39		
0	0	0	0	0	nop	No data-unit operation. Status not modified
0	0	0	0	1	reserved	
0	0	0	1	0	eint	Global-interrupt enable
0	0	0	1	1	dint	Global-interrupt disable
0	0	1	x	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). See Figure 45 for PP opcode formats. 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 local/global (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. 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. See Figure 45 for PP opcode formats.

Table 23. Parallel Transfer Type

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

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.

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 settings 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. See Figure 45 for PP opcode formats.

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 is:

```
(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

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, 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)
- Shift register transfer (SRT) packet transfers for updating VRAM-based frame buffers
- DRAM refresh
- Host bus request

TC functional block diagram

Figure 46 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
  - Programmable:

    bus size: 8-, 16-, 32-, or 64-bits
    page size
    bank size
    address multiplexing
    cycle timing
    block-write mode
    bank priority
  - 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)

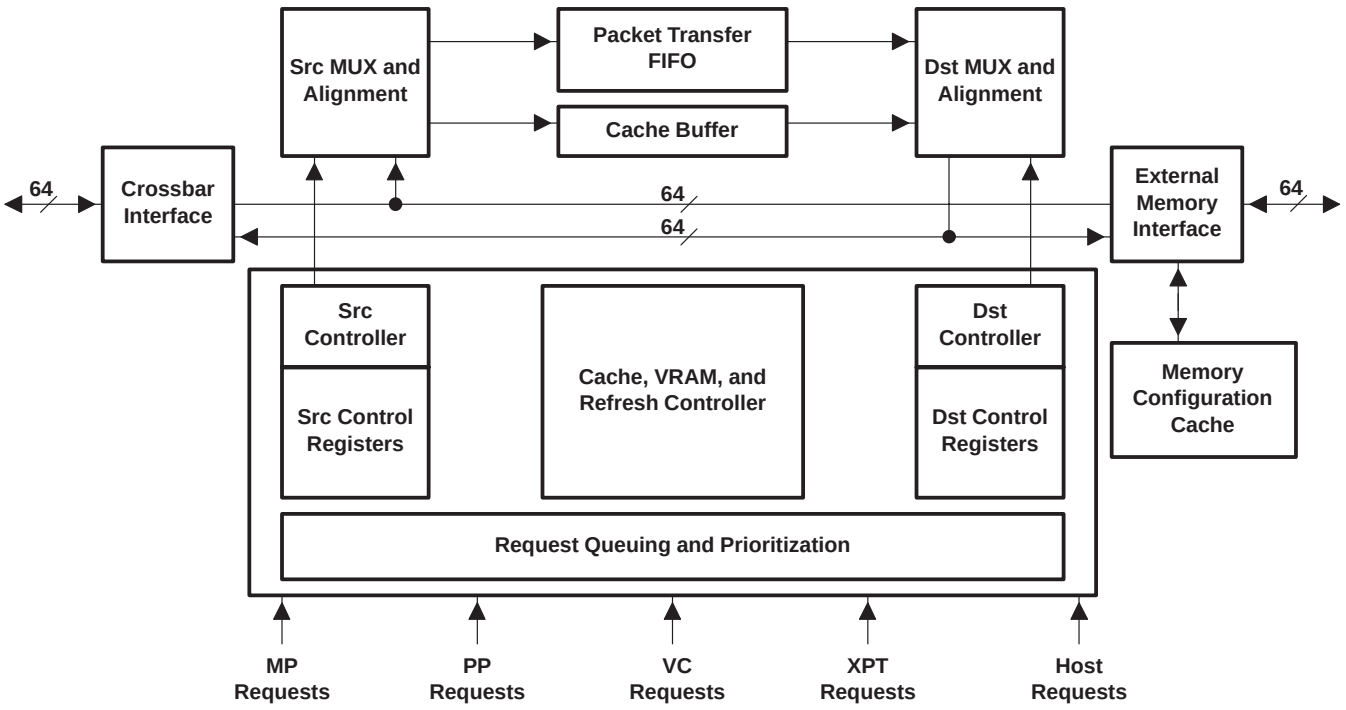


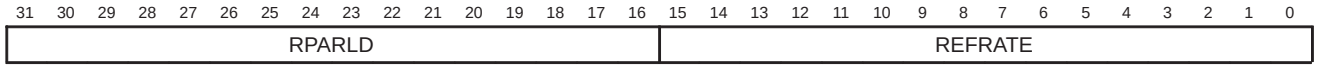
Figure 46. TC Block Diagram

TC registers

The TC contains four on-chip memory-mapped registers accessible by the MP.

refresh control (REFCNTL) register (0x01820000)

The REFCNTL register controls refresh cycles.



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

Figure 47. 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.

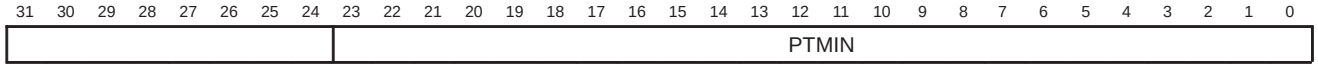


Figure 48. PTMIN Register

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.

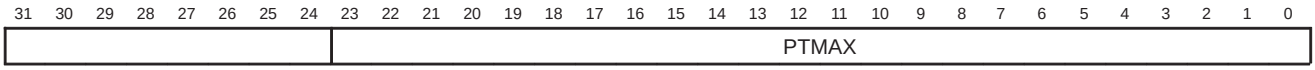


Figure 49. PTMAX Register

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.

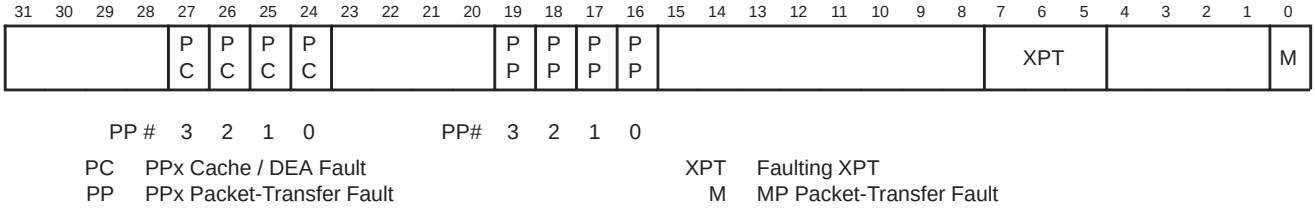


Figure 50. FLTSTS Register

packet-transfer parameters

The most efficient method for data movement in a SMJ320C80 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, 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 TC on the SMJ320C80 supports short- and long-form packet transfers. The PT parameter table format is shown in Figure 51.

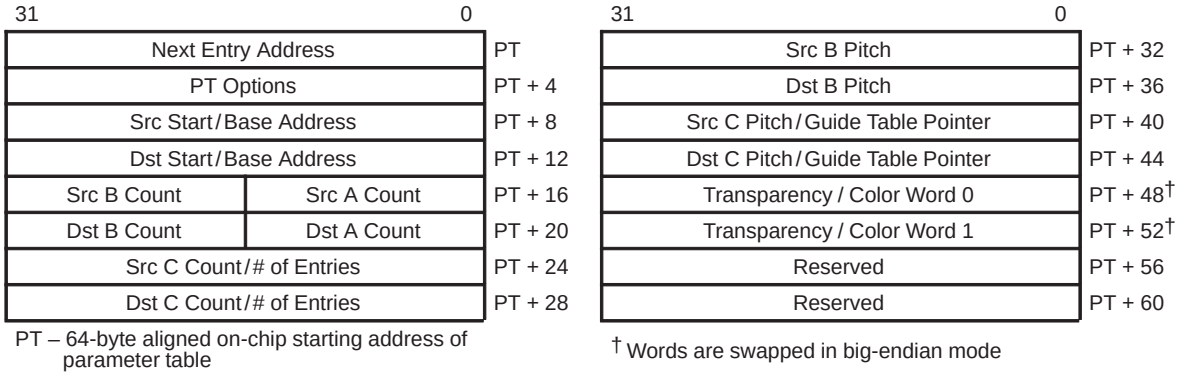
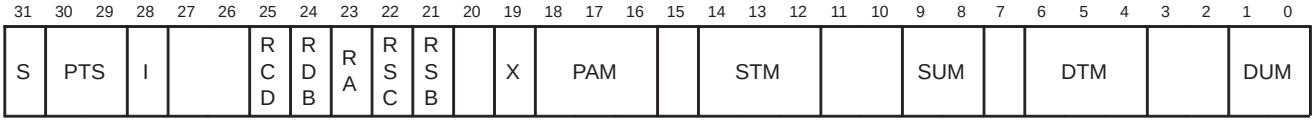


Figure 51. Packet-Transfer Parameter Table

PT-options field

The PT-options field of the parameter table controls the type of src and dst transfer that the TC performs. The format of the options field is shown in Figure 52.



S	Stop bit		PAM	PT Access Mode	
PTS	PT status			000 – Normal	100 – 8 Bit Transfer
	00 – Active	10 – Fault on src		001 – PDT	101 – 16 Bit Transfer
	01 – Suspended	11 – Fault on dst		010 – Block Write	110 – 32 bit Transfer
I	Interrupt when complete			011 – SRT	111 – 64 Bit Transfer
RDC	Reverse dst C addressing		STM/DTM	src/dst Transfer Mode	
RDB	Reverse dst B addressing			000 – Dimensioned	100 – Var Delta-Guided
RA	Reverse A addressing			001 – Fill†	101 – Var Offset-Guided
RSC	Reverse src C addressing			010 – Reserved	110 – Fixed Delta-Guided
RSB	Reverse src B addressing			011 – LUT†	111 – Fixed Offset-Guided
X	Exchange src and dst parameters		SUM/DUM	src/dst update mode	
				00 – None	01 – Add C Pitch
				01 – Add B Pitch	11 – Add C Pitch/Reverse

† Valid for src only.

Figure 52. 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, 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.

Table 29. Row-Time Status Codes

STATUS[5:0]	CYCLE TYPE
0 0 0 0 0 0	Normal Read
0 0 0 0 0 1	Normal Write
0 0 0 0 1 0	Refresh
0 0 0 0 1 1	SDRAM DCAB
0 0 0 1 0 0	Peripheral Device PT Read
0 0 0 1 0 1	Peripheral Device PT Write
0 0 0 1 1 0	Reserved
0 0 0 1 1 1	Reserved
0 0 1 0 0 0	Reserved
0 0 1 0 0 1	Block-Write PT
0 0 1 0 1 0	Reserved
0 0 1 0 1 1	Reserved
0 0 1 1 0 0	SDRAM MRS
0 0 1 1 0 1	Load Color Register
0 0 1 1 1 0	Reserved
0 0 1 1 1 1	Reserved
0 1 0 0 0 0	Frame 0 Read Transfer
0 1 0 0 0 1	Frame 0 Write Transfer
0 1 0 0 1 0	Frame 0 Split-Read Transfer
0 1 0 0 1 1	Frame 0 Split-Write Transfer
0 1 0 1 0 0	Frame 1 Read Transfer
0 1 0 1 0 1	Frame 1 Write Transfer
0 1 0 1 1 0	Frame 1 Split-Read Transfer
0 1 0 1 1 1	Frame 1 Split-Write Transfer
0 1 1 0 0 0	Reserved
0 1 1 0 0 1	Reserved
0 1 1 0 1 0	Reserved
0 1 1 0 1 1	Reserved
0 1 1 1 0 0	PT Read Transfer
0 1 1 1 0 1	PT Write Transfer
0 1 1 1 1 0	Reserved
0 1 1 1 1 1	Idle

STATUS[5:0]	CYCLE TYPE
1 0 0 0 0 0	Reserved
1 0 0 0 0 1	Reserved
1 0 0 0 1 0	Reserved
1 0 0 0 1 1	Reserved
1 0 0 1 0 0	XPT1 Read
1 0 0 1 0 1	XPT1 Write
1 0 0 1 1 0	XPT1 PDPT Read
1 0 0 1 1 1	XPT1 PDPT Write
1 0 1 0 0 0	XPT2 Read
1 0 1 0 0 1	XPT2 Write
1 0 1 0 1 0	XPT2 PDPT Read
1 0 1 0 1 1	XPT2 PDPT Write
1 0 1 1 0 0	XPT3 Read
1 0 1 1 0 1	XPT3 Write
1 0 1 1 1 0	XPT3 PDPT Read
1 0 1 1 1 1	XPT3 PDPT Write
1 1 0 0 0 0	XPT4/SAM1 Read
1 1 0 0 0 1	XPT4/SAM1 Write
1 1 0 0 1 0	XPT4/SAM1 PDPT Read
1 1 0 0 1 1	XPT4/SAM1 PDPT Write
1 1 0 1 0 0	XPT5/SOF1 Read
1 1 0 1 0 1	XPT5/SOF1 Write
1 1 0 1 1 0	XPT5/SOF1 PDPT Read
1 1 0 1 1 1	XPT5/SOF1 PDPT Write
1 1 1 0 0 0	XPT6/SAM0 Read
1 1 1 0 0 1	XPT6/SAM0 Write
1 1 1 0 1 0	XPT6/SAM0 PDPT Read
1 1 1 0 1 1	XPT6/SAM0 PDPT Write
1 1 1 1 0 0	XPT7/SOF0 Read
1 1 1 1 0 1	XPT7/SOF0 Write
1 1 1 1 1 0	XPT7/SOF0 PDPT Read
1 1 1 1 1 1	XPT7/SOF0 PDPT Write

local memory interface (continued)

Table 30. Column-Time Status Codes

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

STATUS[5:0]	CYCLE TYPE
1 0 0 0 0 0	Reserved
1 0 0 0 0 1	Reserved
1 0 0 0 1 0	Reserved
1 0 0 0 1 1	Reserved
1 0 0 1 0 0	Reserved
1 0 0 1 0 1	Reserved
1 0 0 1 1 0	Reserved
1 0 0 1 1 1	Reserved
1 0 1 0 0 0	Reserved
1 0 1 0 0 1	Reserved
1 0 1 0 1 0	Reserved
1 0 1 0 1 1	Reserved
1 0 1 1 0 0	Reserved
1 0 1 1 0 1	Reserved
1 0 1 1 1 0	Reserved
1 0 1 1 1 1	Reserved
1 1 0 0 0 0	Reserved
1 1 0 0 0 1	Reserved
1 1 0 0 1 0	Reserved
1 1 0 0 1 1	Reserved
1 1 0 1 0 0	Reserved
1 1 0 1 0 1	Reserved
1 1 0 1 1 0	Reserved
1 1 0 1 1 1	Reserved
1 1 1 0 0 0	Reserved
1 1 1 0 0 1	Reserved
1 1 1 0 1 0	Reserved
1 1 1 0 1 1	Reserved
1 1 1 1 0 0	Reserved
1 1 1 1 0 1	Reserved
1 1 1 1 1 0	Reserved
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 SMJ320C80 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.

A 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 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
100	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
011	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
100	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 53. Address Multiplexing

dynamic bus sizing

The ’C80 supports data bus sizes of 8, 16, 32, or 64 bits. 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 $\overline{\text{CAS}}$ 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] pins at row time. The selected timing remains in effect until the next row access.

Table 32. Cycle-Timing Selection

CT[2:0]	MEMORY TIMING
0 0 0	Pipelined (Burst Length 1) SDRAM, CAS Latency of 2
0 0 1	Pipelined (Burst Length 1) SDRAM, CAS Latency of 3
0 1 0	Interleaved (Burst Length 2) SDRAM, CAS Latency of 2
0 1 1	Interleaved (Burst Length 2) SDRAM, CAS Latency of 3
1 0 0	Pipelined 1 Cycle/Column
1 0 1	Nonpipelined 1 Cycle/Column
1 1 0	2 Cycle/Column
1 1 1	3 Cycle/Column

page sizing

Whenever an external memory access occurs, the TC records the 22 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.

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 SMJ320C80 supports three modes of VRAM block-write. The block-write mode is dynamically selectable so that software can specify block-writes regardless of the 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.

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 SMJ320C80 provides direct support for synchronous DRAM (SDRAM), synchronous VRAM (SVRAM), and synchronous 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 54 shows the MRS value generated by the ’C80.

SDRAM Mode Register Bit	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	1	CT0	0	0	0	CT1

CT0, CT1 as input at the start of the MRS cycle

Figure 54. 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 55). 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 55. MRS Value Alignment

memory cycles

SMJ320C80 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 56. Memory cycles consist of row states and the column pipeline.

memory cycles (continued)

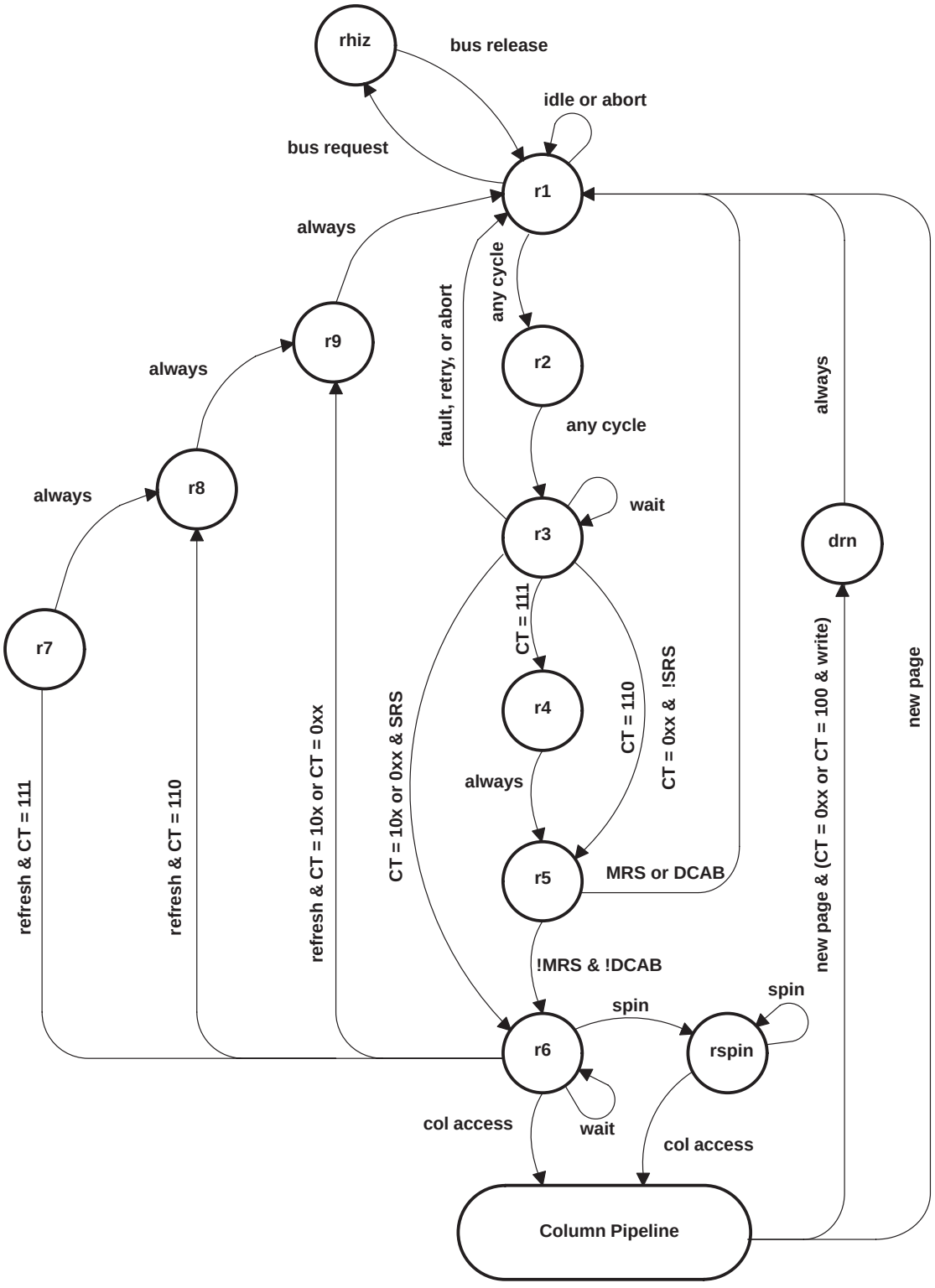


Figure 56. Memory Cycle State Diagram

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.

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 $\overline{RL}$ and drives $\overline{DDIN}$ according to the data transfer direction. AS[2:0], BS[1:0], CT[2:0], PS[3:0], and $\overline{UTIME}$ inputs are sampled
r3	Common to all memory accesses. $\overline{DBEN}$ is driven to its active level. For non-SDRAM, $\overline{W}$, $\overline{TRG}$/$\overline{CAS}$, and $\overline{DSF}$ are driven to their active levels, and for non-SDRAM refreshes, all $\overline{CAS}$/DQM strobes are activated. $\overline{FAULT}$, READY, and $\overline{RETRY}$ inputs are sampled.
r4	Inserted for 3 cycle/column accesses (CT=111) only. No signal transitions occur. $\overline{RETRY}$ input is sampled.
r5	Common to SDRAM and 2 or 3 cycle/column accesses (CT=0xx or 11x). $\overline{RAS}$ is driven low. $\overline{W}$ is driven low for DCAB and MRS cycles and $\overline{TRG}$/$\overline{CAS}$ is driven low for MRS and SDRAM refresh cycles.
r6	Common to all memory accesses. For SDRAM cycles, $\overline{RAS}$, $\overline{TRG}$/$\overline{CAS}$, and $\overline{W}$ are driven high. For non-SDRAM, $\overline{RAS}$ is driven low (if not already) and $\overline{W}$, $\overline{TRG}$ / $\overline{CAS}$, and DSF are driven to their appropriate levels. $\overline{DBEN}$ is driven low and READY and $\overline{RETRY}$ are sampled.
rspin	Additional state to allow TC column time pipeline to load. No signal transitions occur. $\overline{RETRY}$ 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]. $\overline{RETRY}$ input is sampled.
r8	For 3 cycle/column refreshes only (CT=111). No signal transitions occur. $\overline{RETRY}$ input is sampled.
r9	Common to all refresh cycles. Processor activity code is output on STATUS[5:0] and $\overline{RETRY}$ input is sampled.
drn	Occurs for SDRAM cycles (CT = 0xx) and pipelined 1 cycle/column writes only. For SDRAM cycles, $\overline{RAS}$, and $\overline{W}$ are activated to perform a DCAB command. For pipelined writes, all $\overline{CAS}$/DQM strobes are activated.
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	$\overline{FAULT}$ input sampled low (in r3 state), memory access faulted
retry	$\overline{RETRY}$ 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 can 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 57 show the pipeline stages for each access type and when the $\overline{\text{CAS}}$/DQM signal corresponding to the column access is activated.

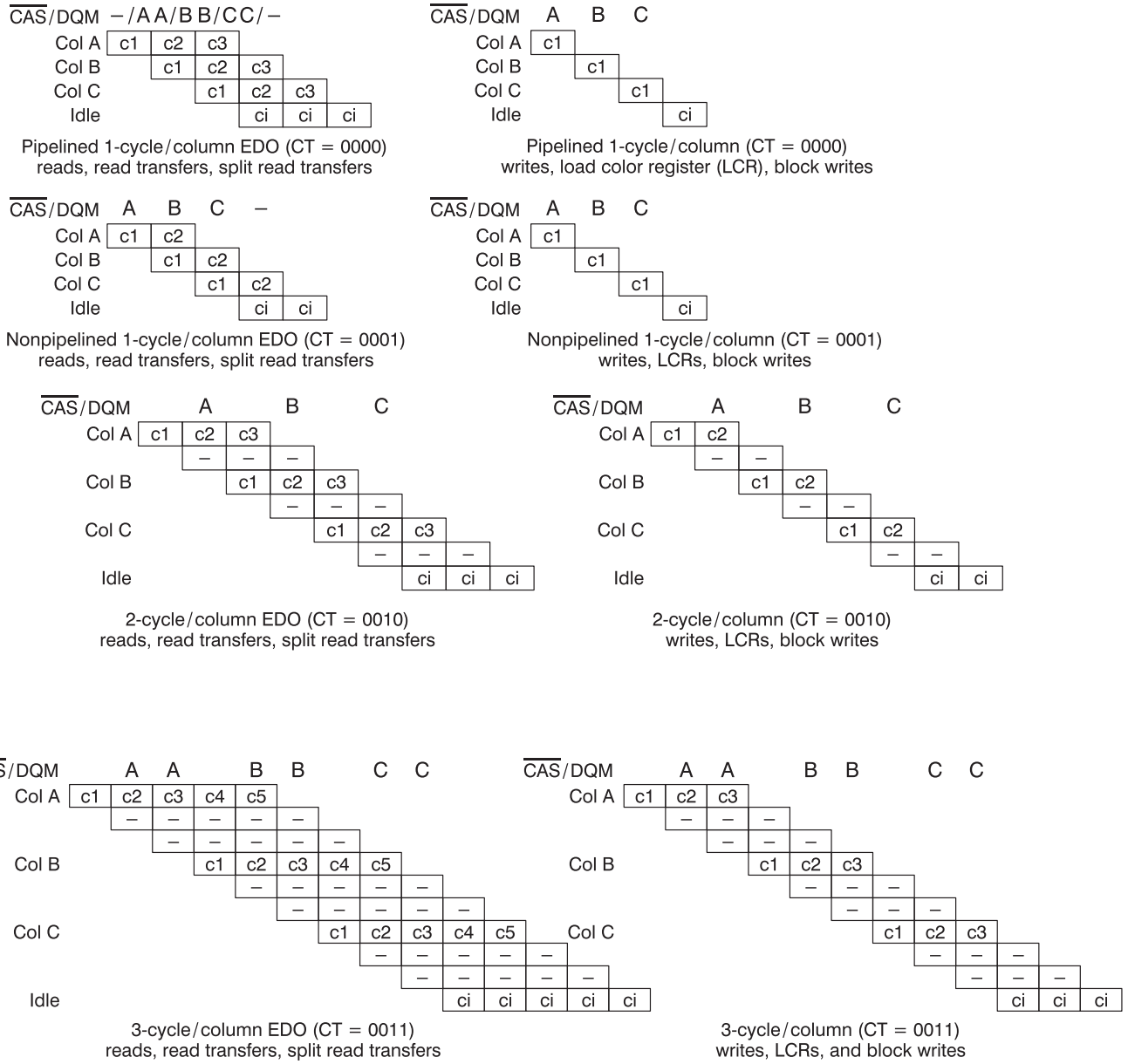


Figure 57. DRAM Cycle Column Pipelines

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, $\overline{W}$ is held high, $\overline{TRG}/\overline{CAS}$ is driven low after $\overline{RAS}$ to enable memory output drivers and $\overline{DBEN}$ and $\overline{DDIN}$ are low so that data transceivers can drive into the ’C80. During column time, 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. During peripheral device packet transfers, $\overline{DBEN}$ and $\overline{DDIN}$ remain high.

read cycles (continued)

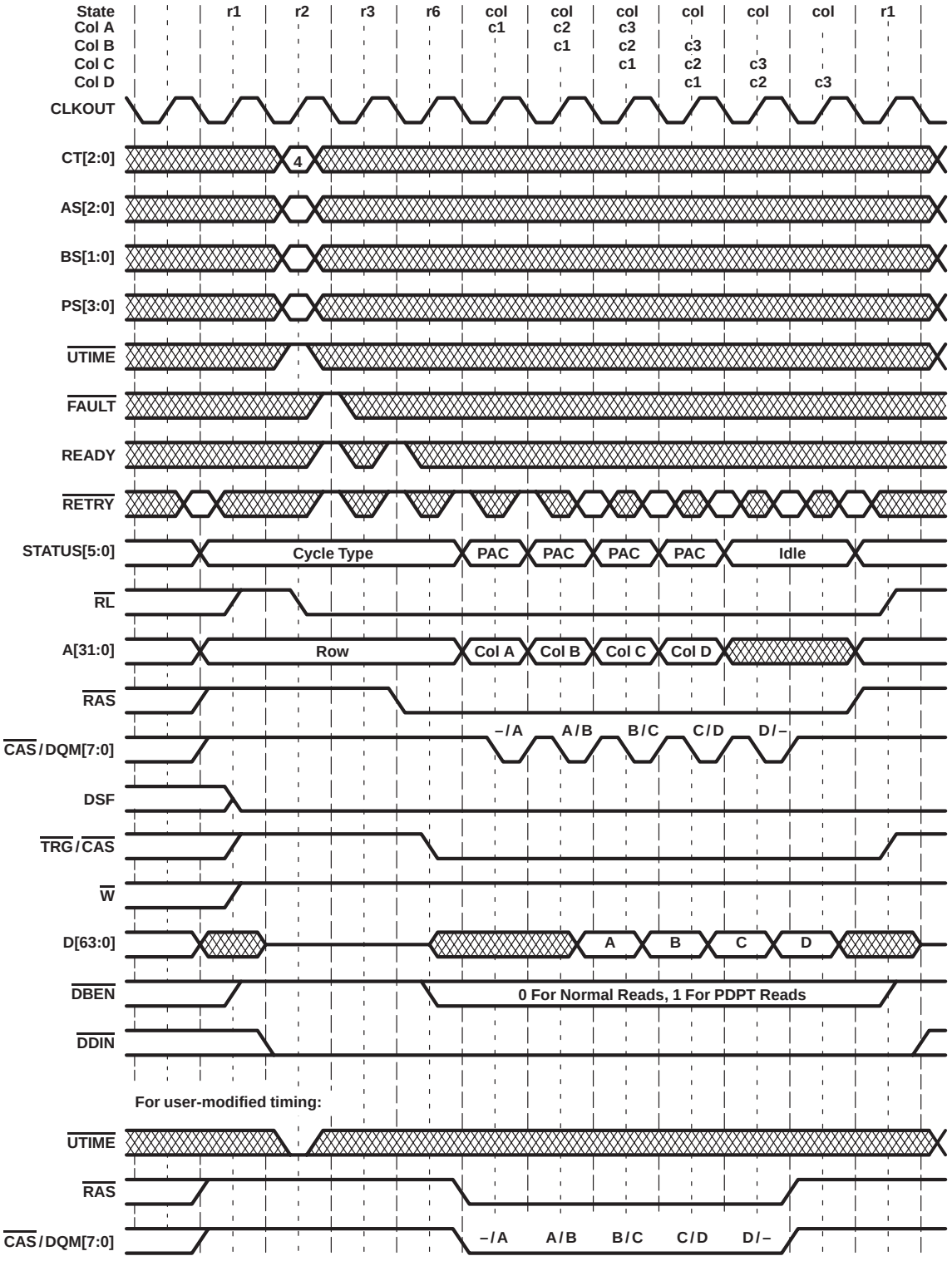


Figure 58. Pipelined 1-Cycle/Column Read-Cycle Timing

read cycles (continued)

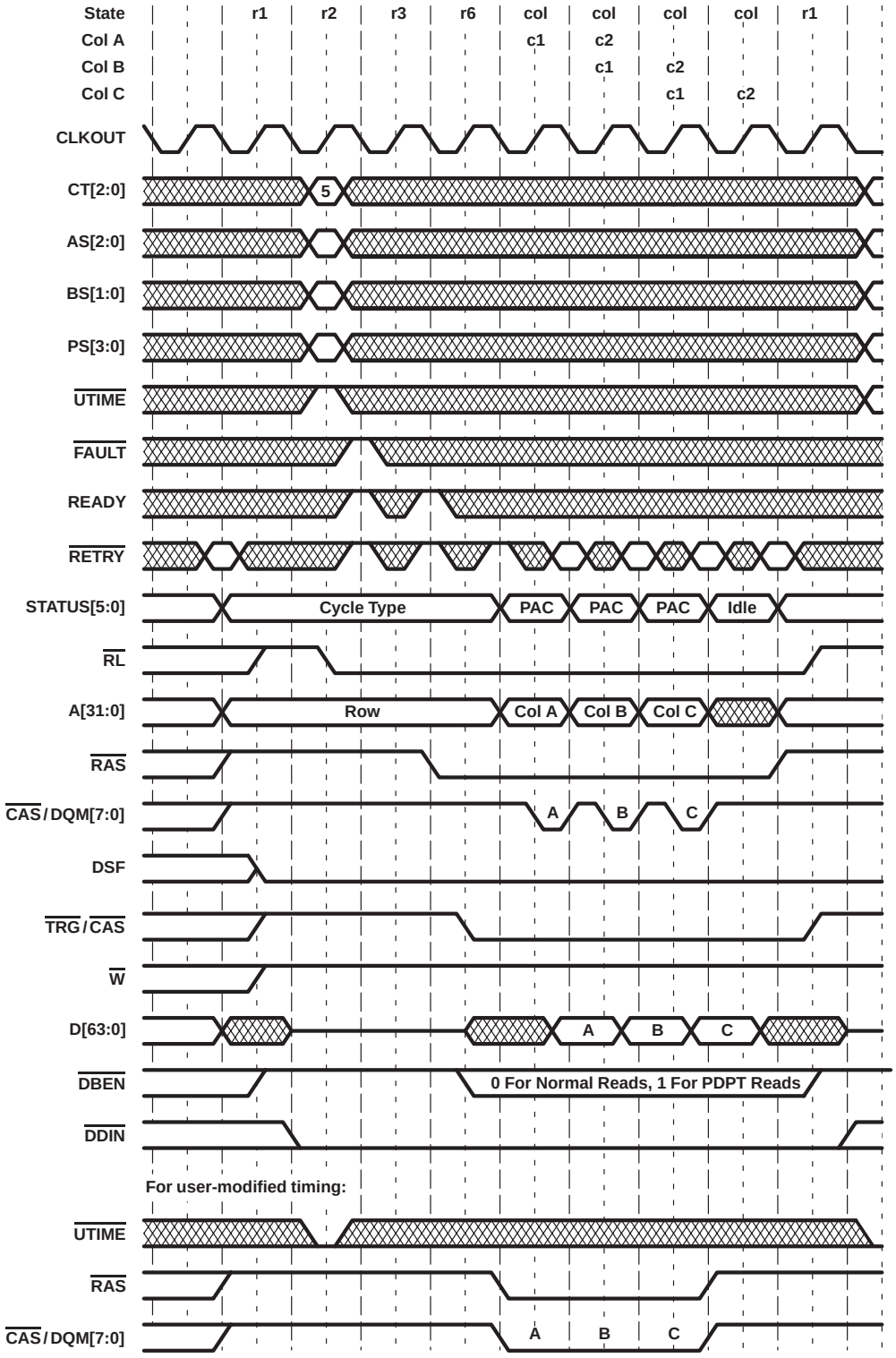
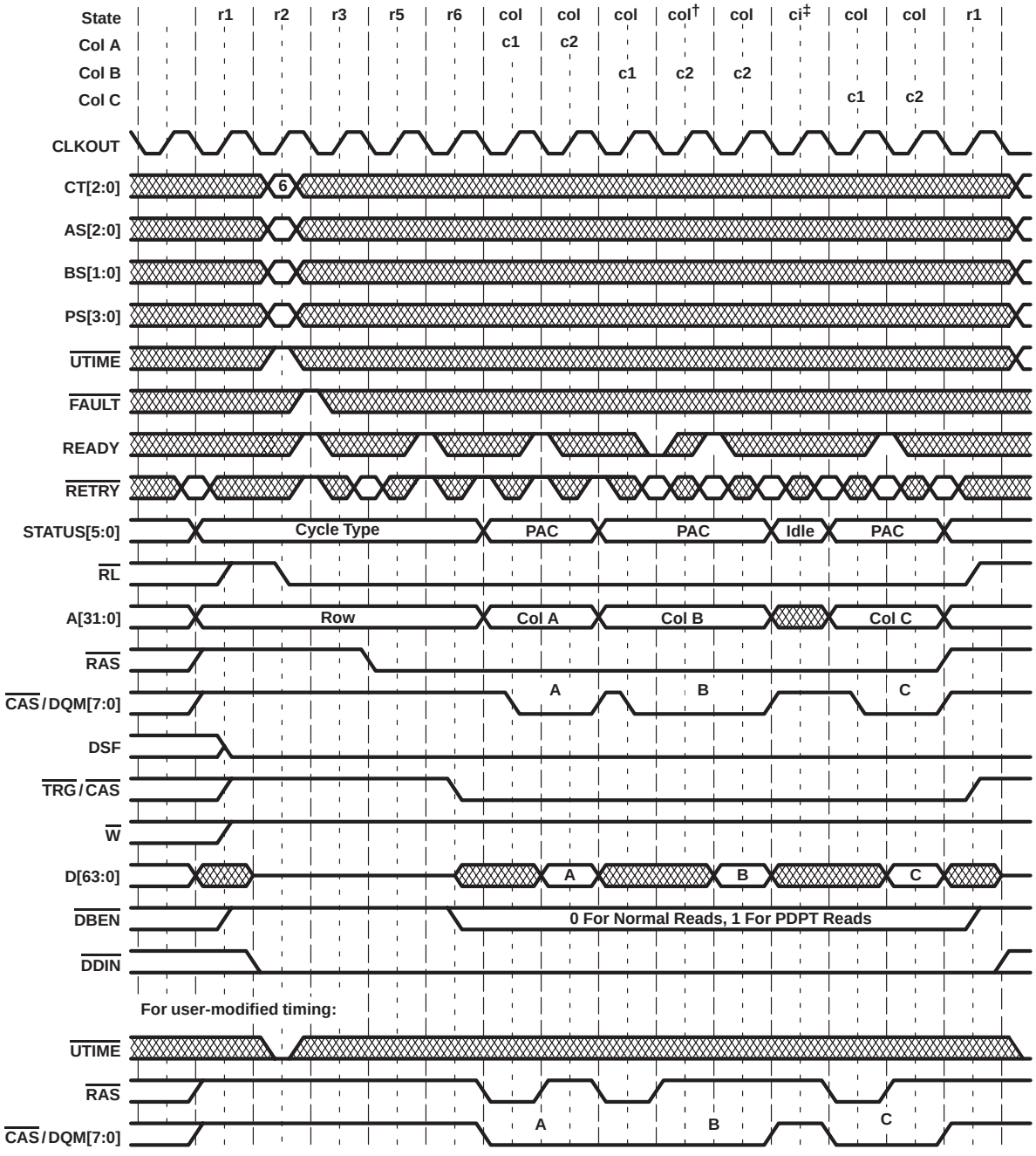


Figure 59. Nonpipelined 1-Cycle/Column Read-Cycle Timing

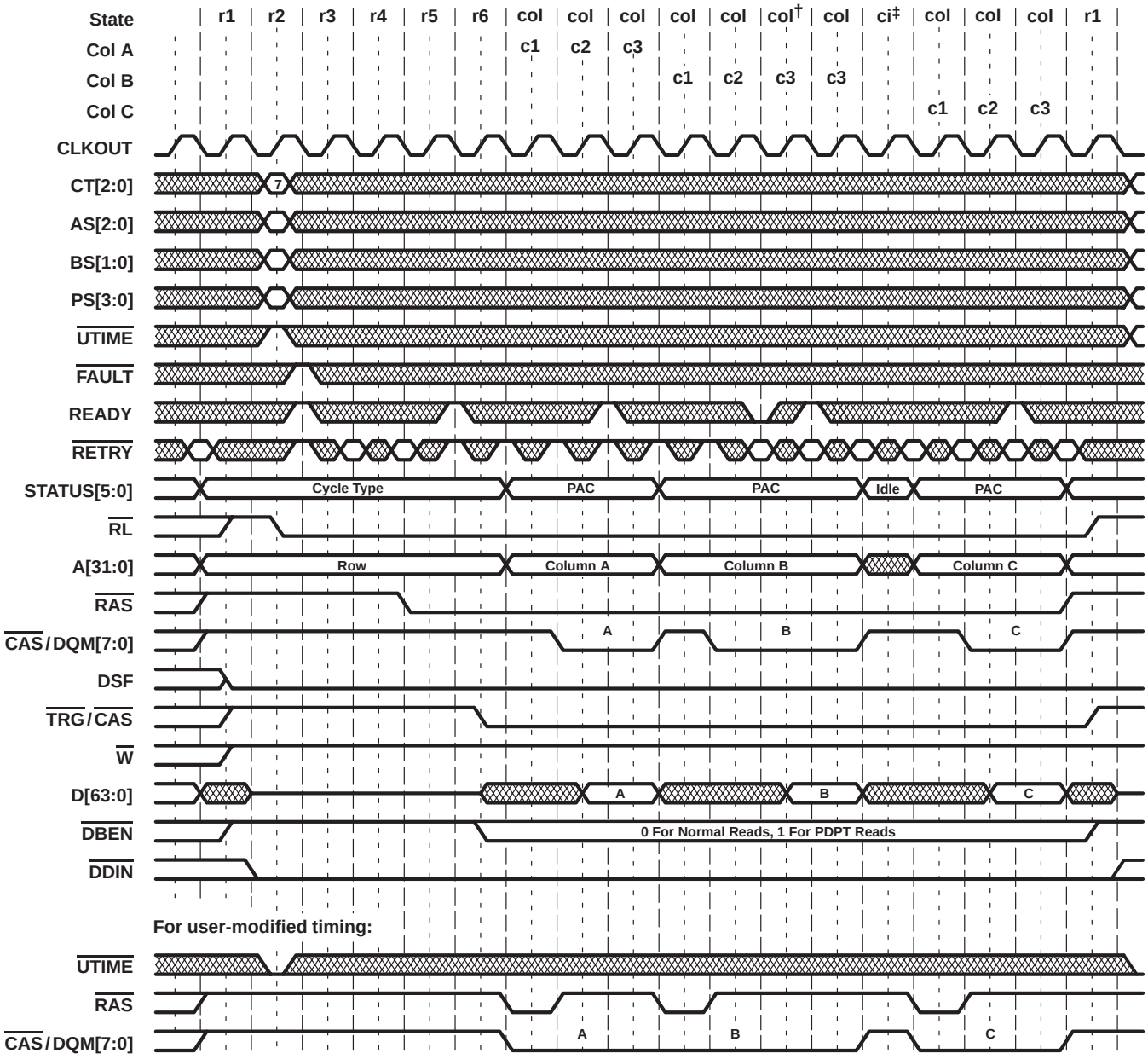
read cycles (continued)



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

Figure 60. 2-Cycle/Column Read-Cycle Timing

read cycles (continued)



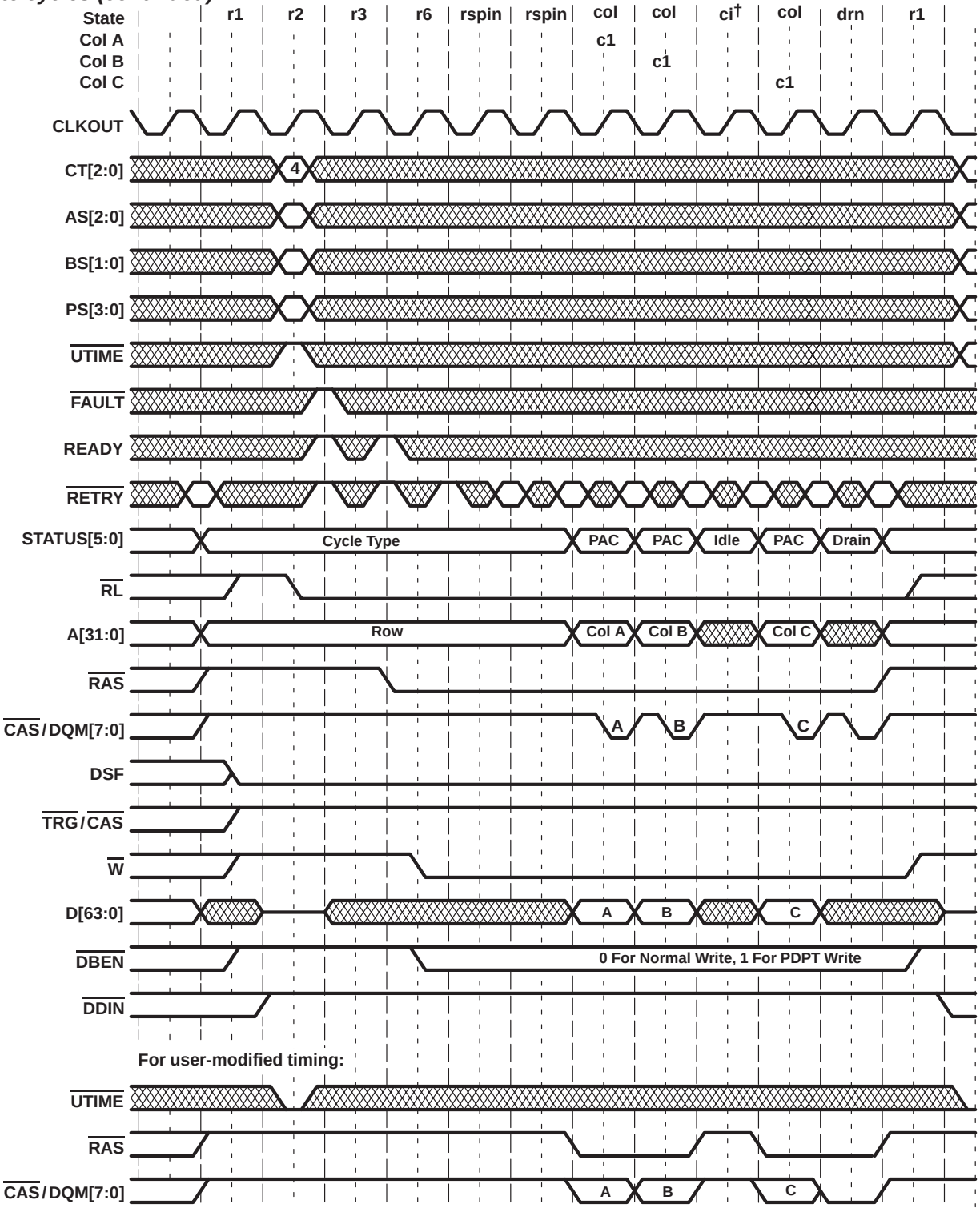
† Wait state inserted by external logic (example)
‡ Internally generated pipeline bubble (example)

Figure 61. 3-Cycle/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.

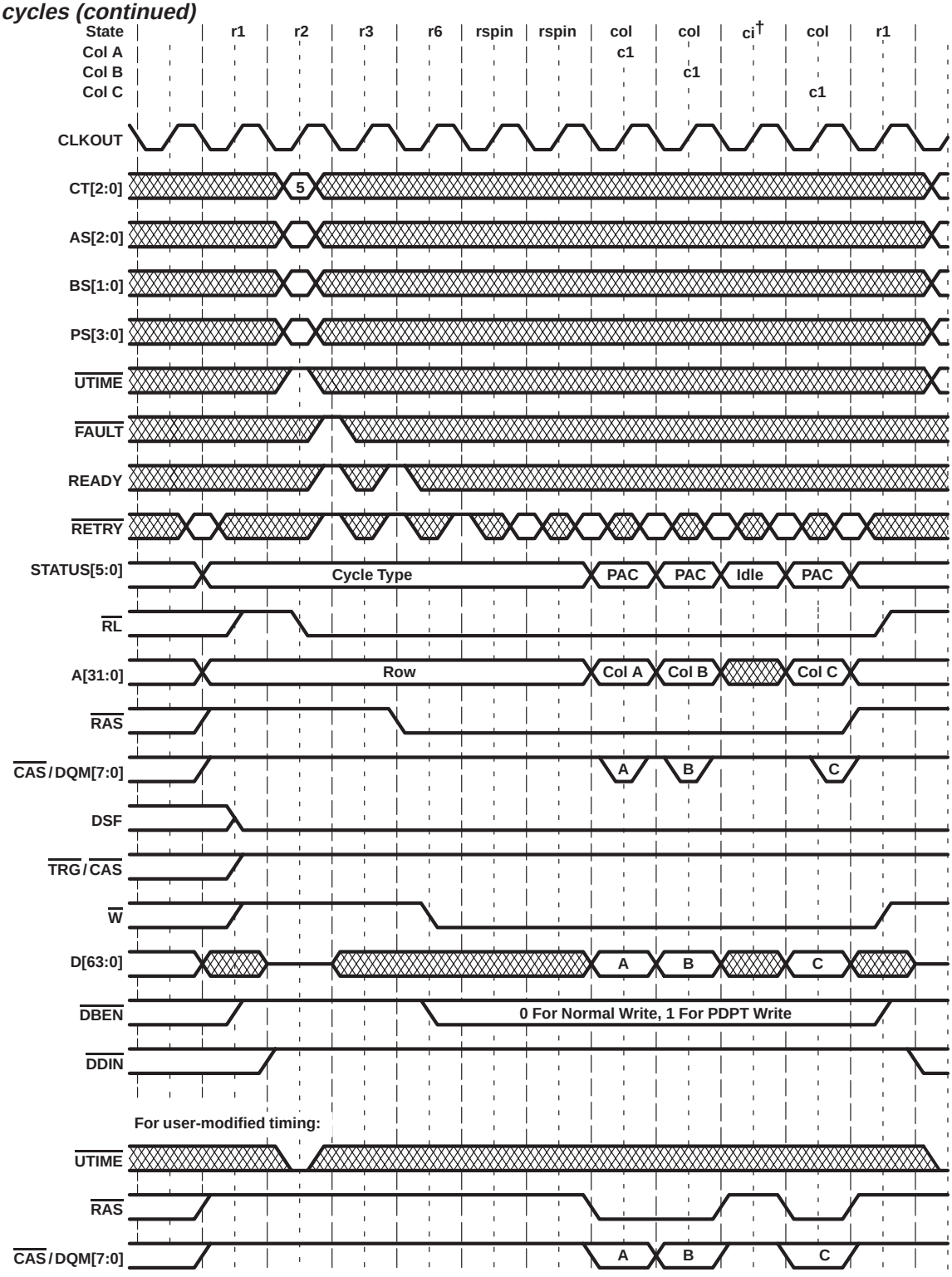
write cycles (continued)



† Internally generated pipeline bubble (example)

Figure 62. Pipelined 1-Cycle/Column Write-Cycle Timing

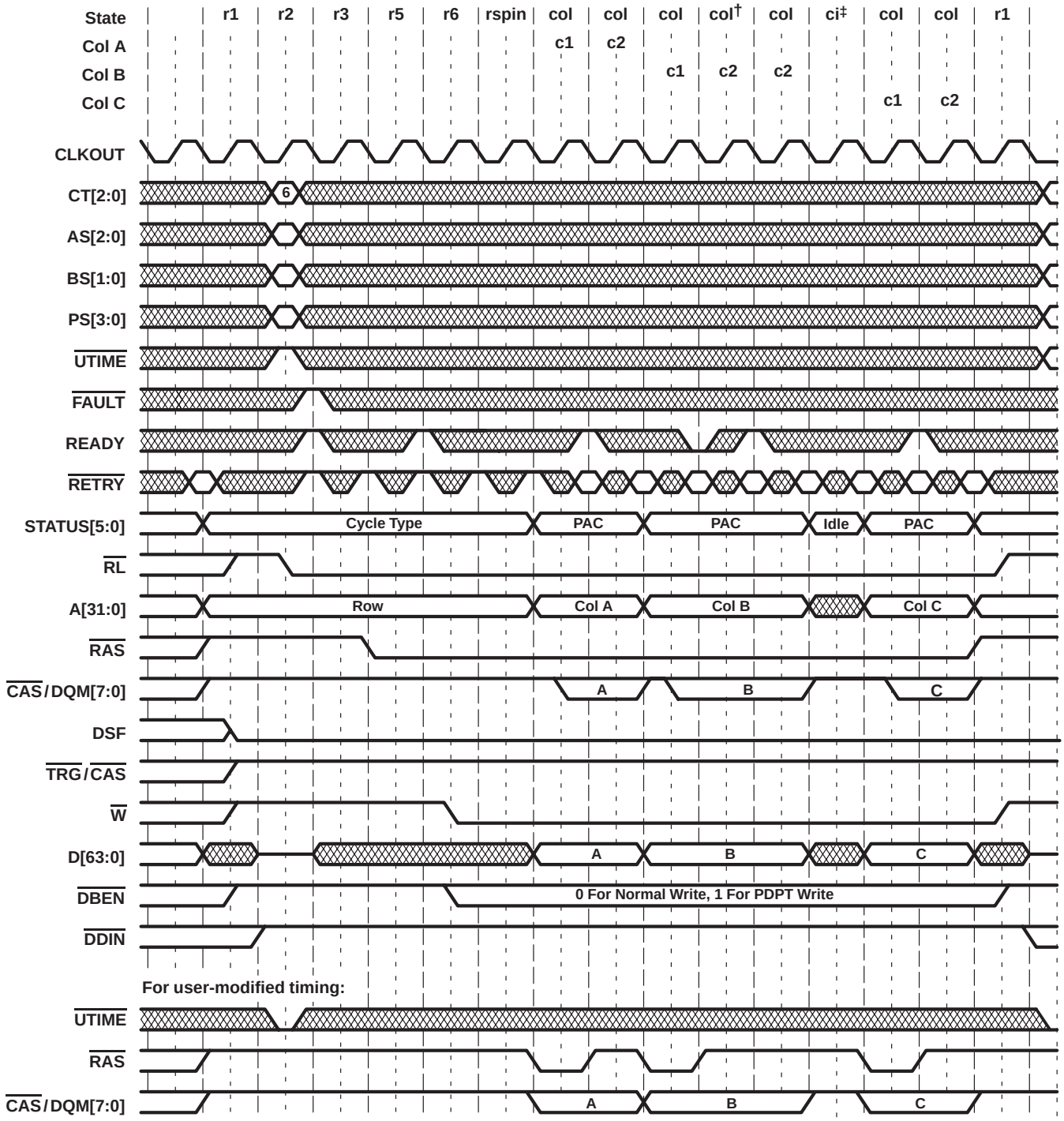
write cycles (continued)



† Internally generated pipeline bubble (example)

Figure 63. Nonpipelined 1-Cycle/Column Write-Cycle Timing

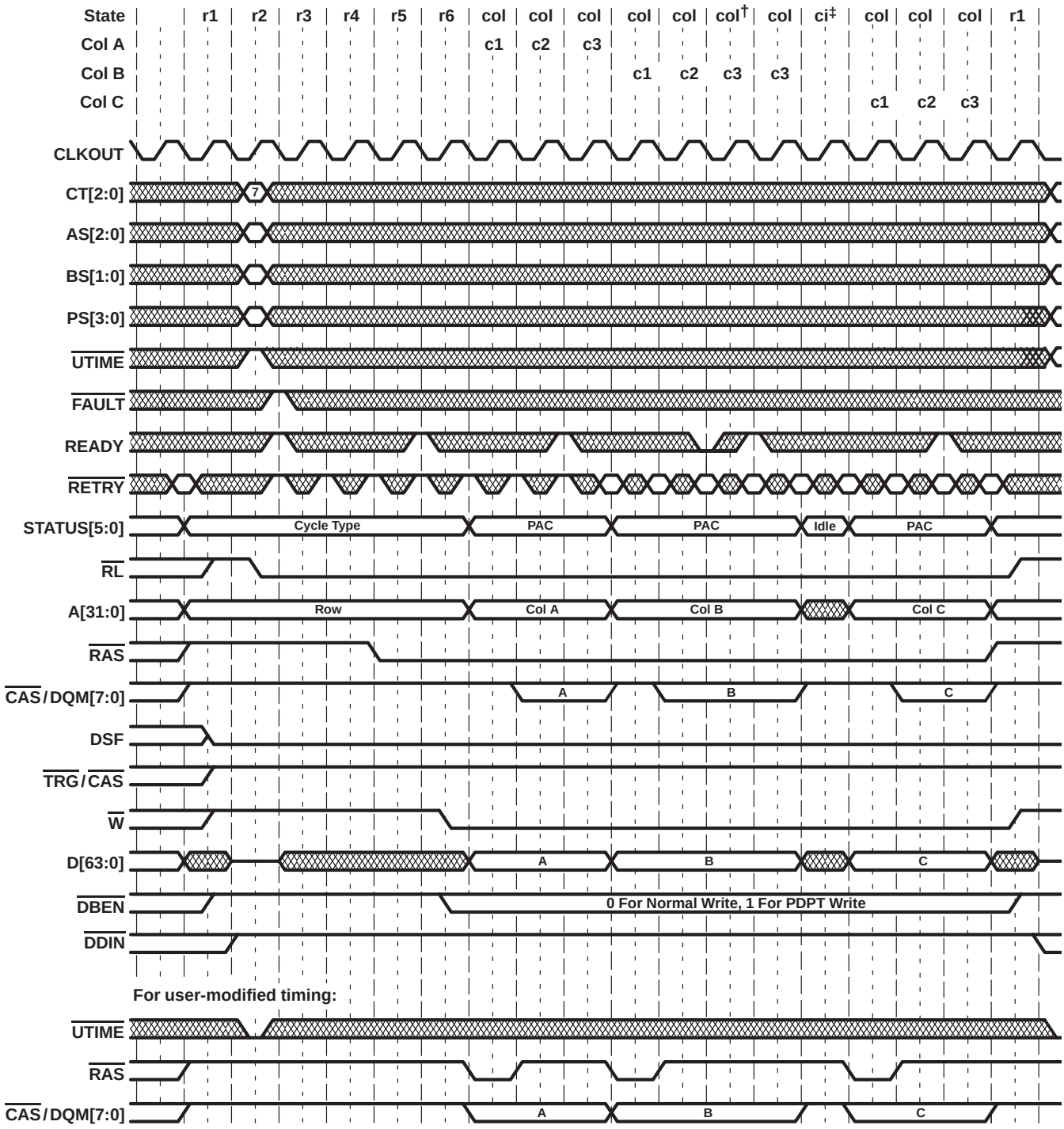
write cycles (continued)



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

Figure 64. 2-Cycle/Column Write-Cycle Timing

write cycles (continued)



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

Figure 65. 3-Cycle/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 $\overline{RAS}$ and the fall of $\overline{CAS}$/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 cannot 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 $\overline{CAS}$/DQM strobes are active during an LCR cycle.

If exception support for a given bank is enabled, the $\overline{EXCEPT}$ [1:0] inputs are sampled during LCR column states and must be at valid levels. A retry code ($\overline{EXCEPT}$ [1:0] = 10) at column time has no effect, however, because only one column access is performed.

If the BW field of the configuration cache entry for the given bank indicates 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 (continued)

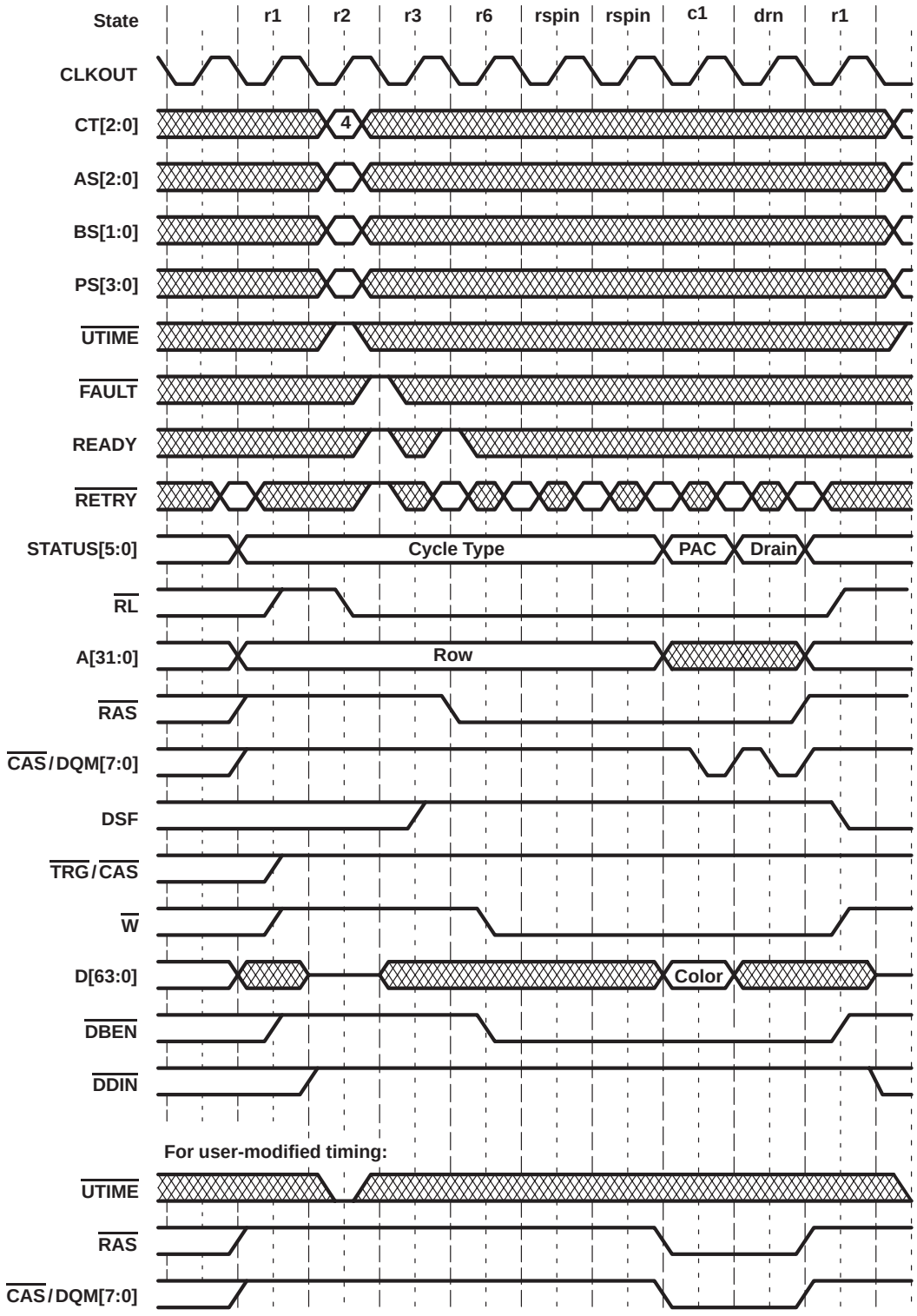


Figure 66. Pipelined 1-Cycle/Column Load-Color-Register-Cycle Timing

load-color-register cycles (continued)

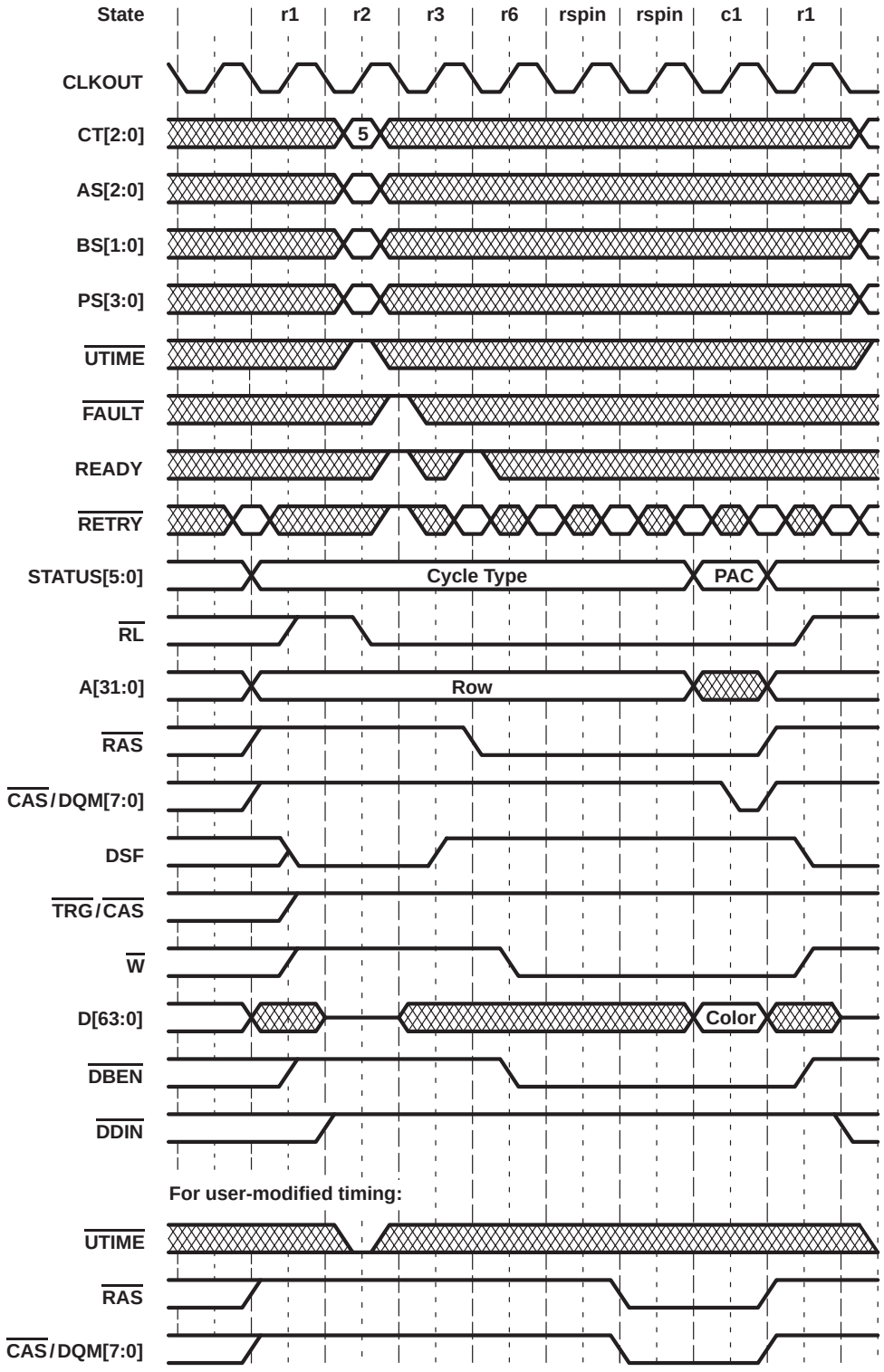


Figure 67. Nonpipelined 1-Cycle/Column Load-Color-Register-Cycle Timing

load-color-register cycles (continued)

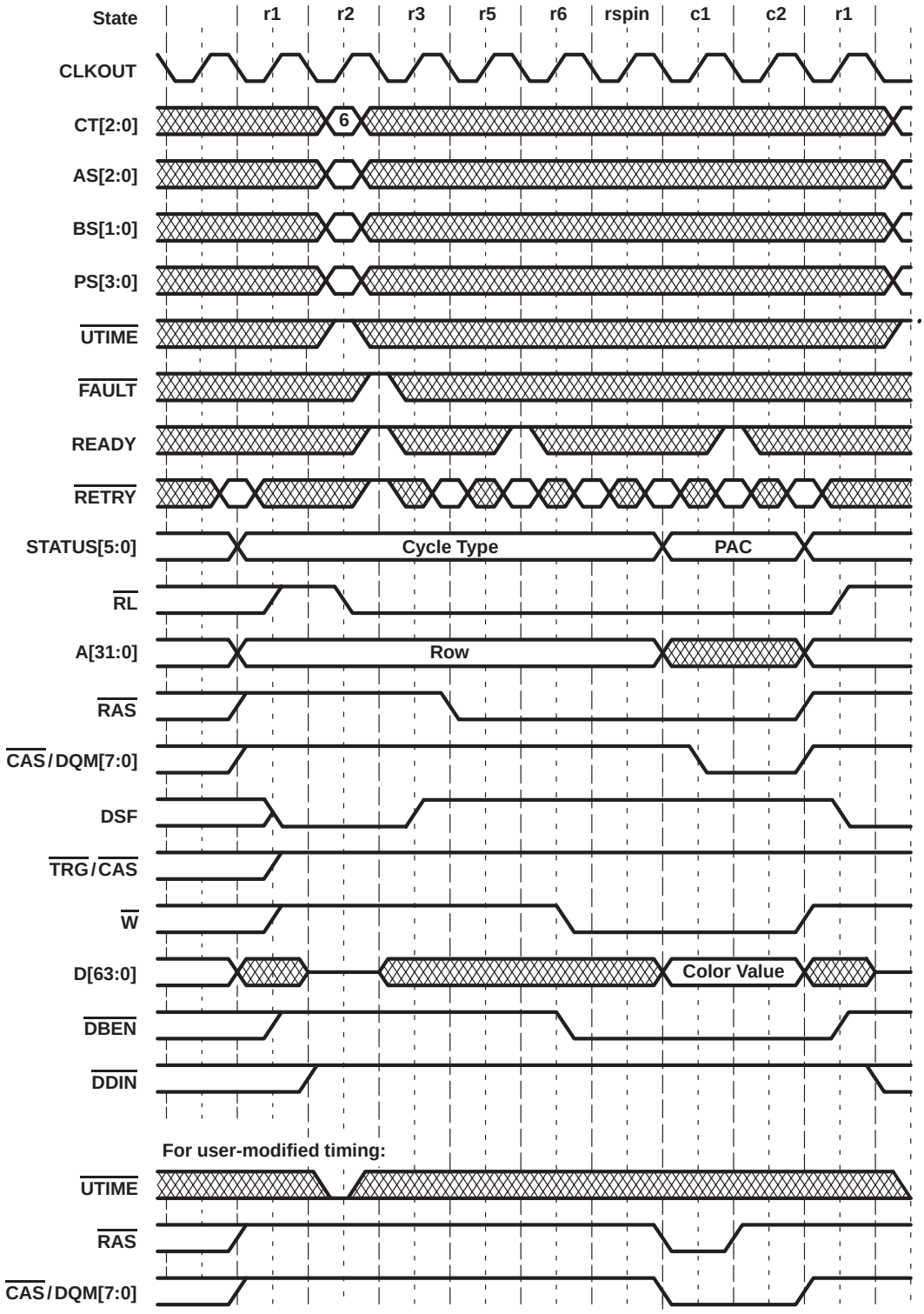


Figure 68. 2-Cycle/Column Load-Color-Register-Cycle Timing

load-color-register cycles (continued)

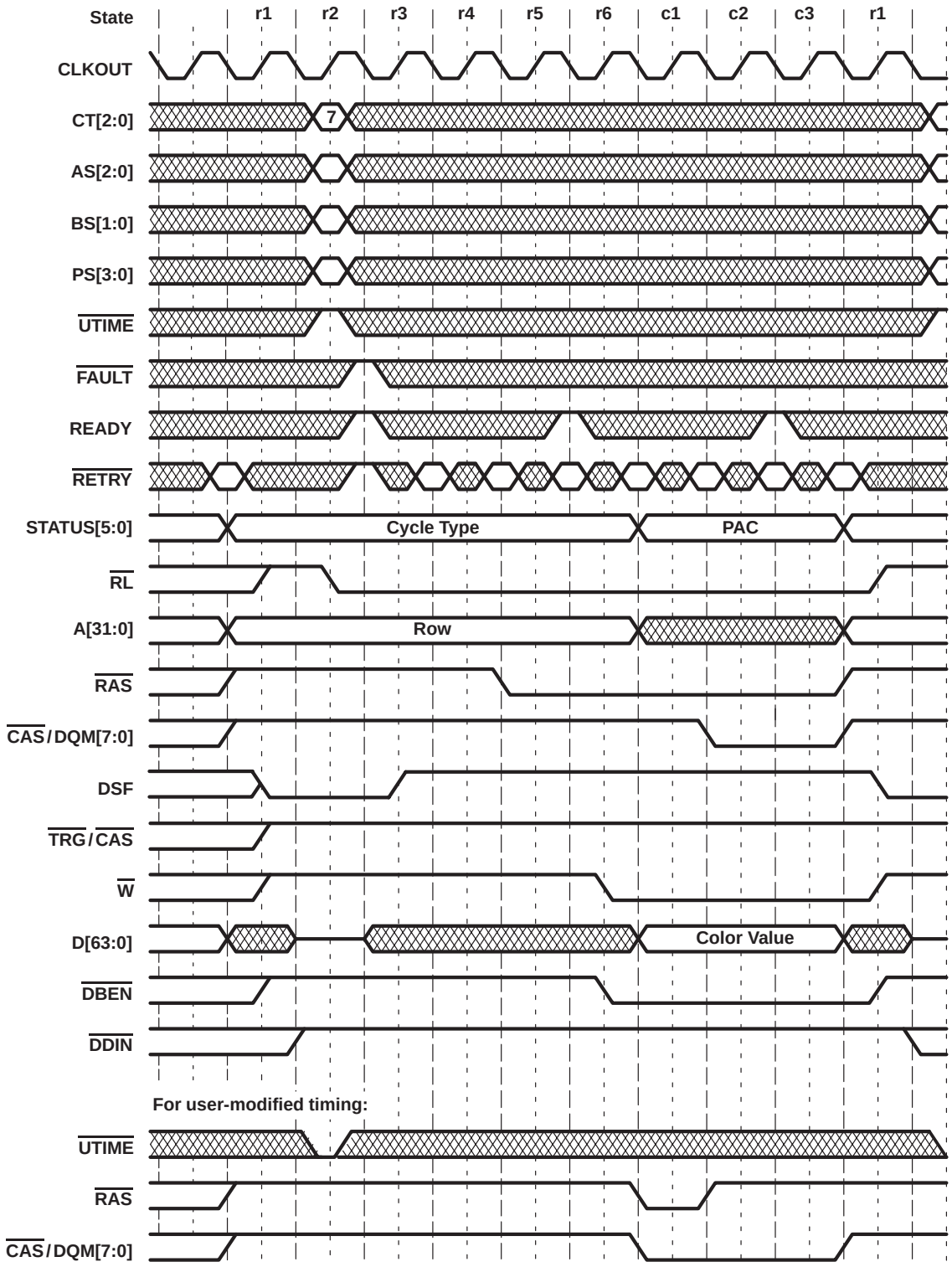


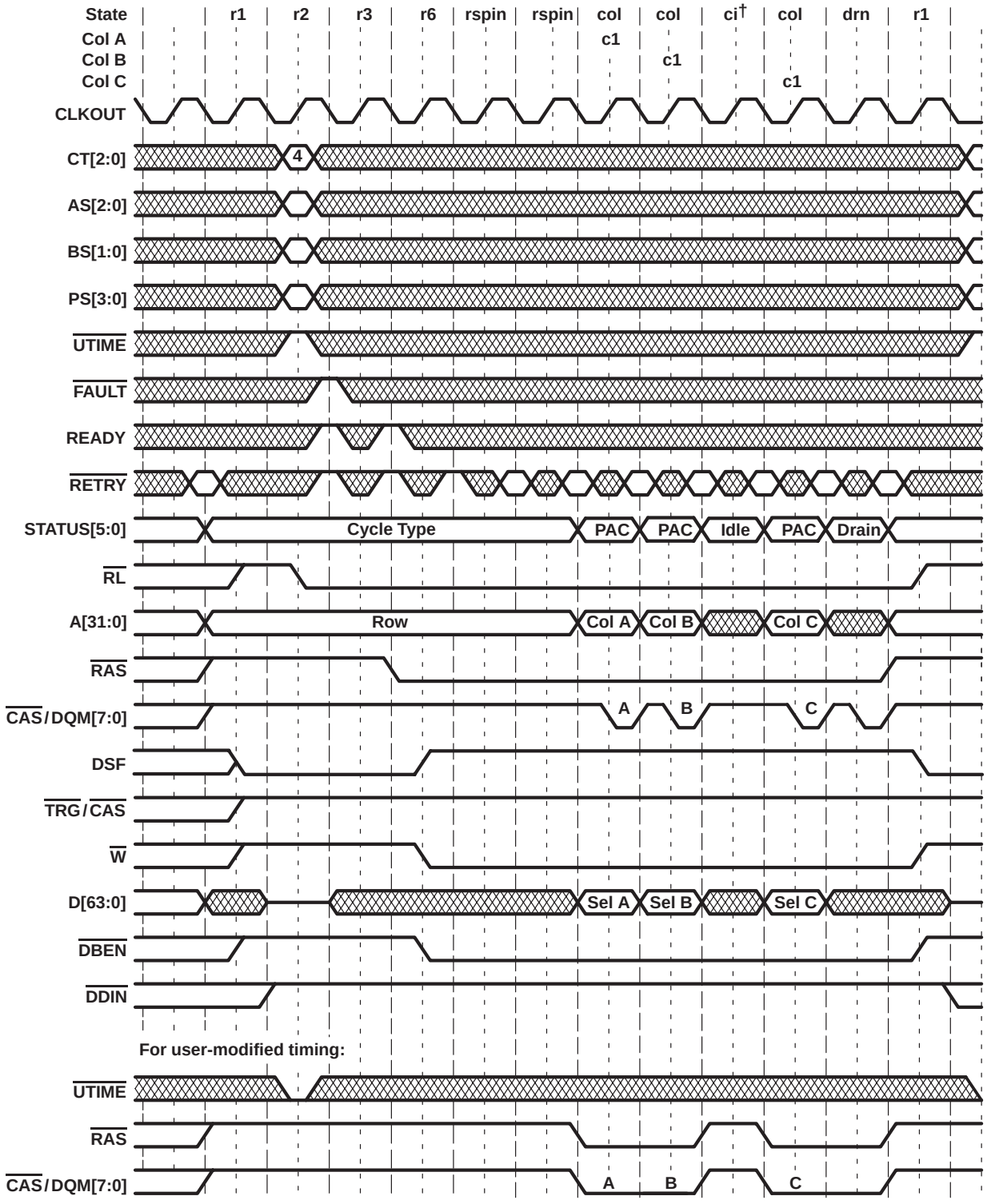
Figure 69. 3-Cycle/Column Load-Color-Register-Cycle Timing

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.

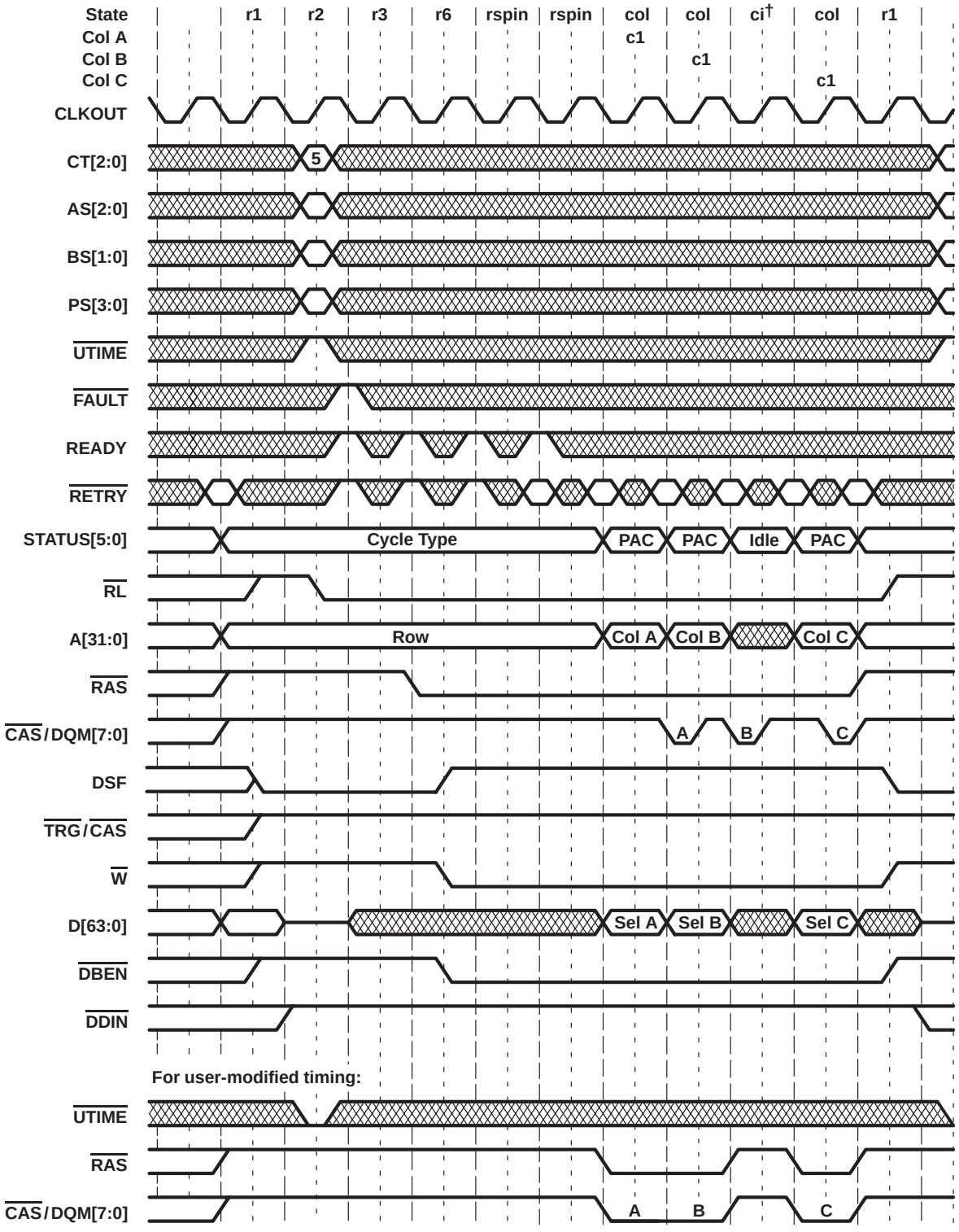
block-write cycles (continued)



† Internally generated pipeline bubble (example)

Figure 70. Pipelined 1-Cycle/Column Block-Write-Cycle Timing

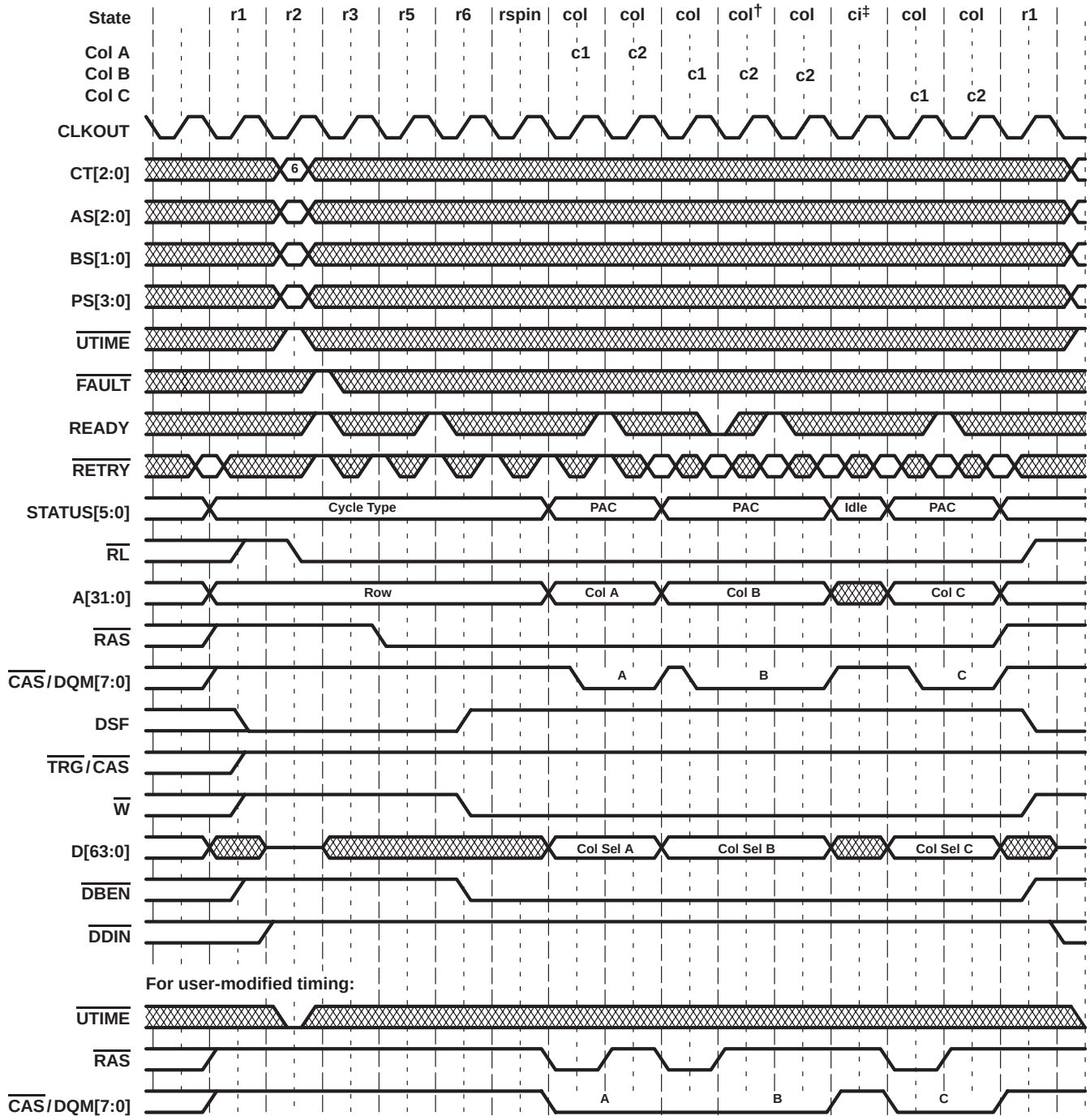
block-write cycles (continued)



† Internally generated pipeline bubble (example)

Figure 71. Nonpipelined 1-Cycle/Column Block-Write-Cycle Timing

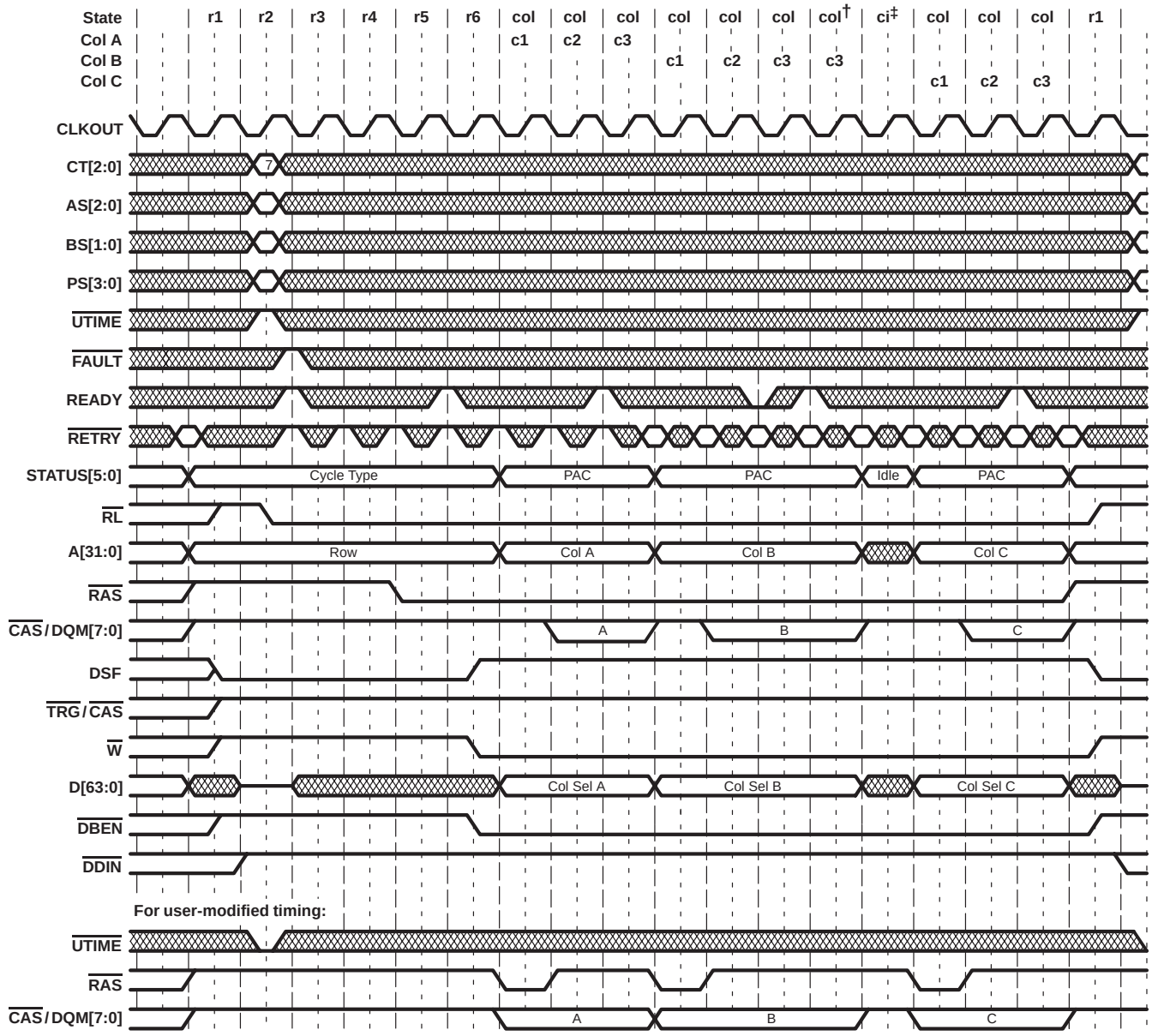
block-write cycles (continued)



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

Figure 72. 2-Cycle/Column Block-Write-Cycle Timing

block-write cycles (continued)



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

Figure 73. 3-Cycle / 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 (sequential-access memory, or 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.

transfer cycles (continued)

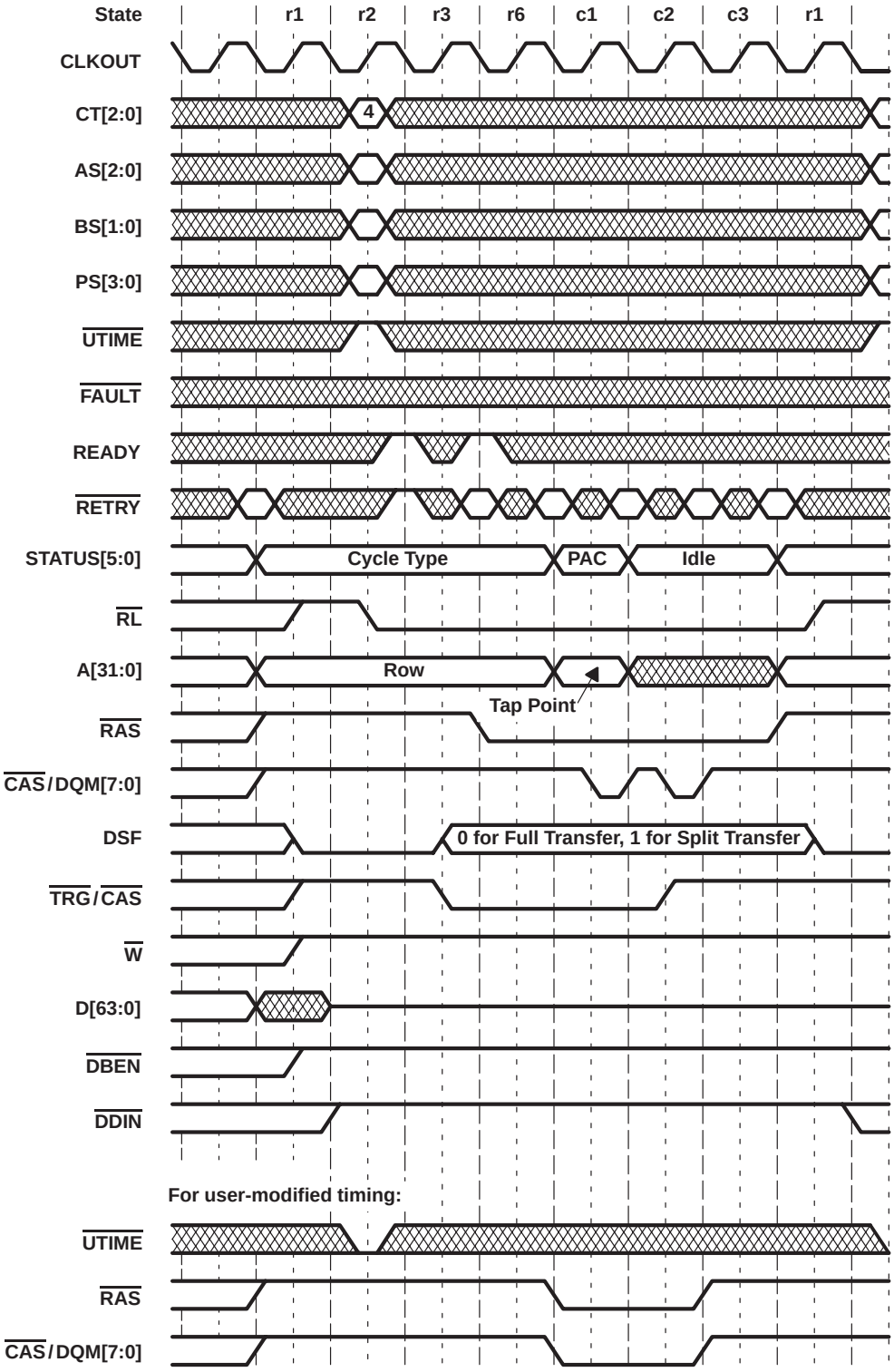


Figure 74. Pipelined 1-Cycle/Column Read-Transfer and Split-Register Read-Transfer-Cycle Timing

transfer cycles (continued)

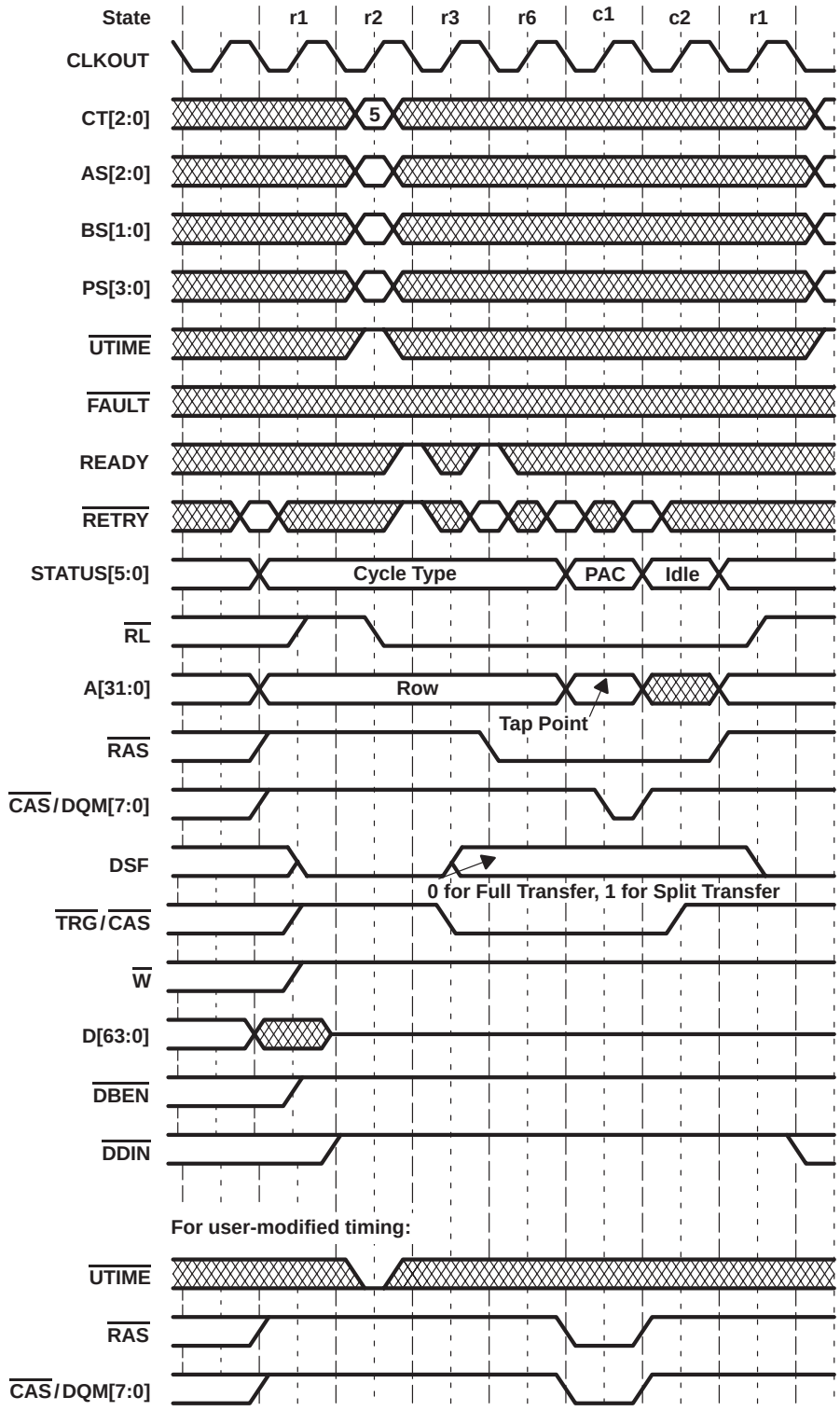


Figure 75. Nonpipelined 1-Cycle/Column Read-Transfer and Split-Register Read-Transfer-Cycle Timing

transfer cycles (continued)

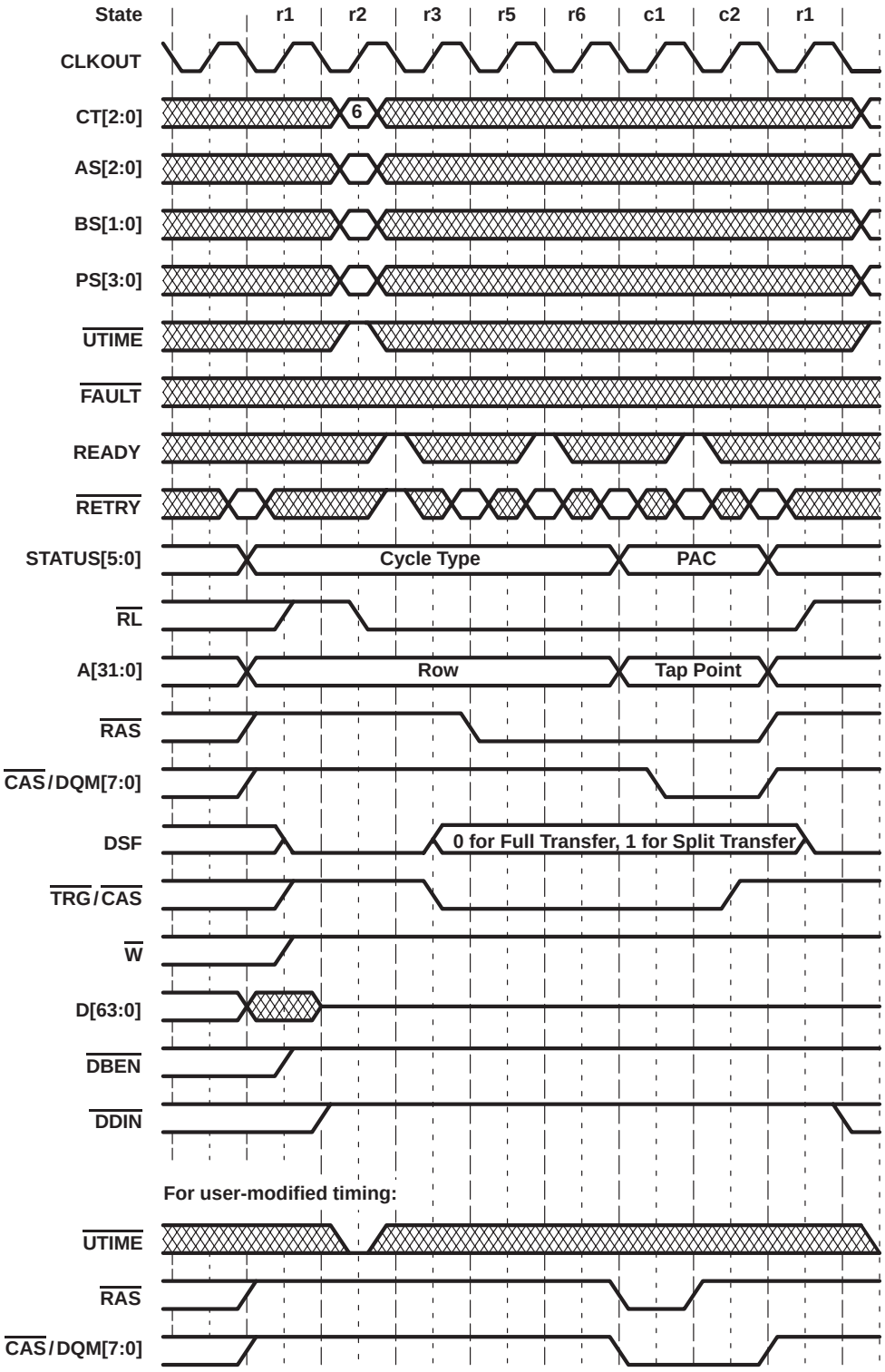


Figure 76. 2-Cycle/Column Read-Transfer and Split-Register Read-Transfer-Cycle Timing

transfer cycles (continued)

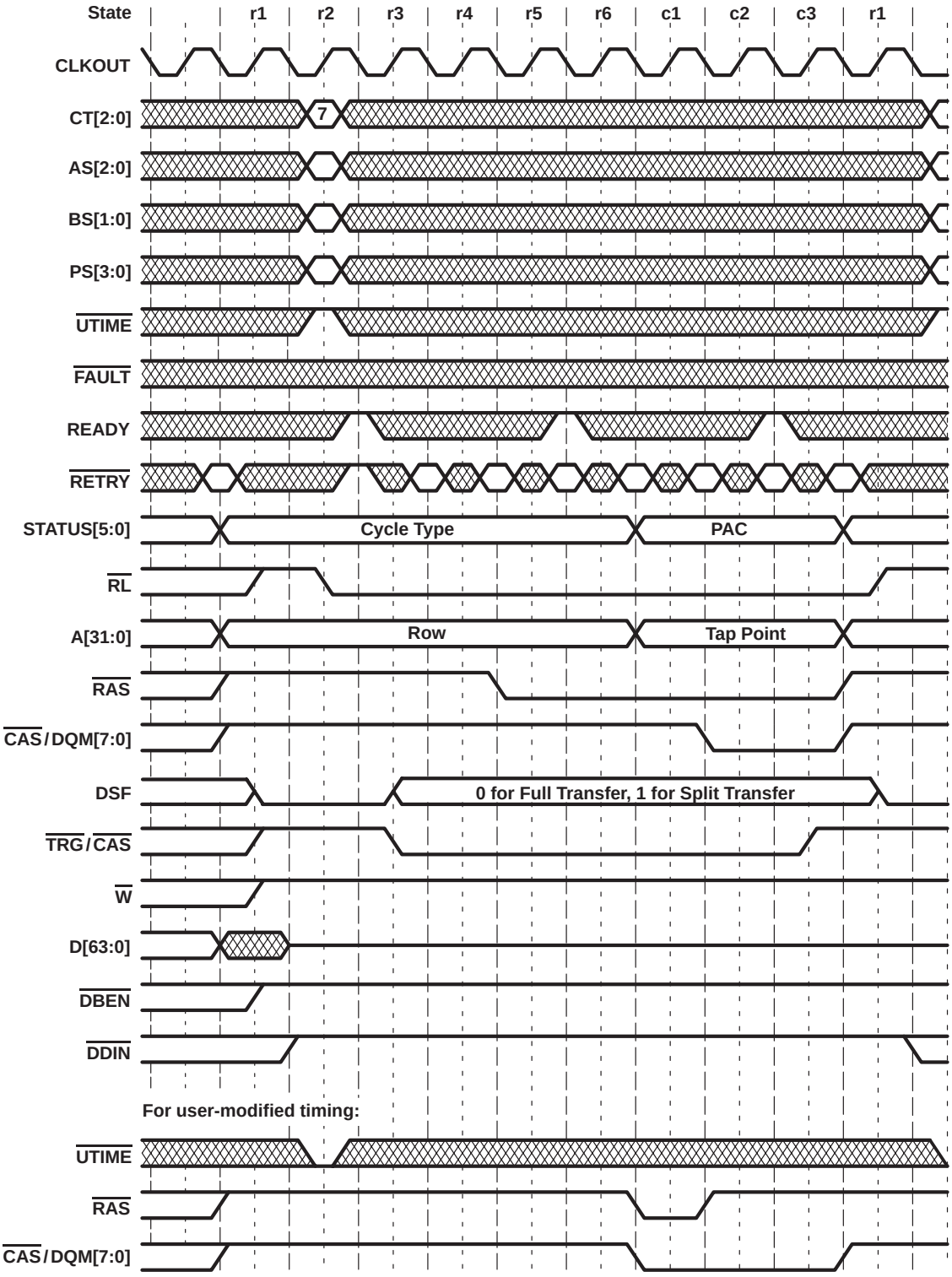


Figure 77. 3-Cycle/Column Read-Transfer and Split-Register Read-Transfer-Cycle Timing

transfer cycles (continued)

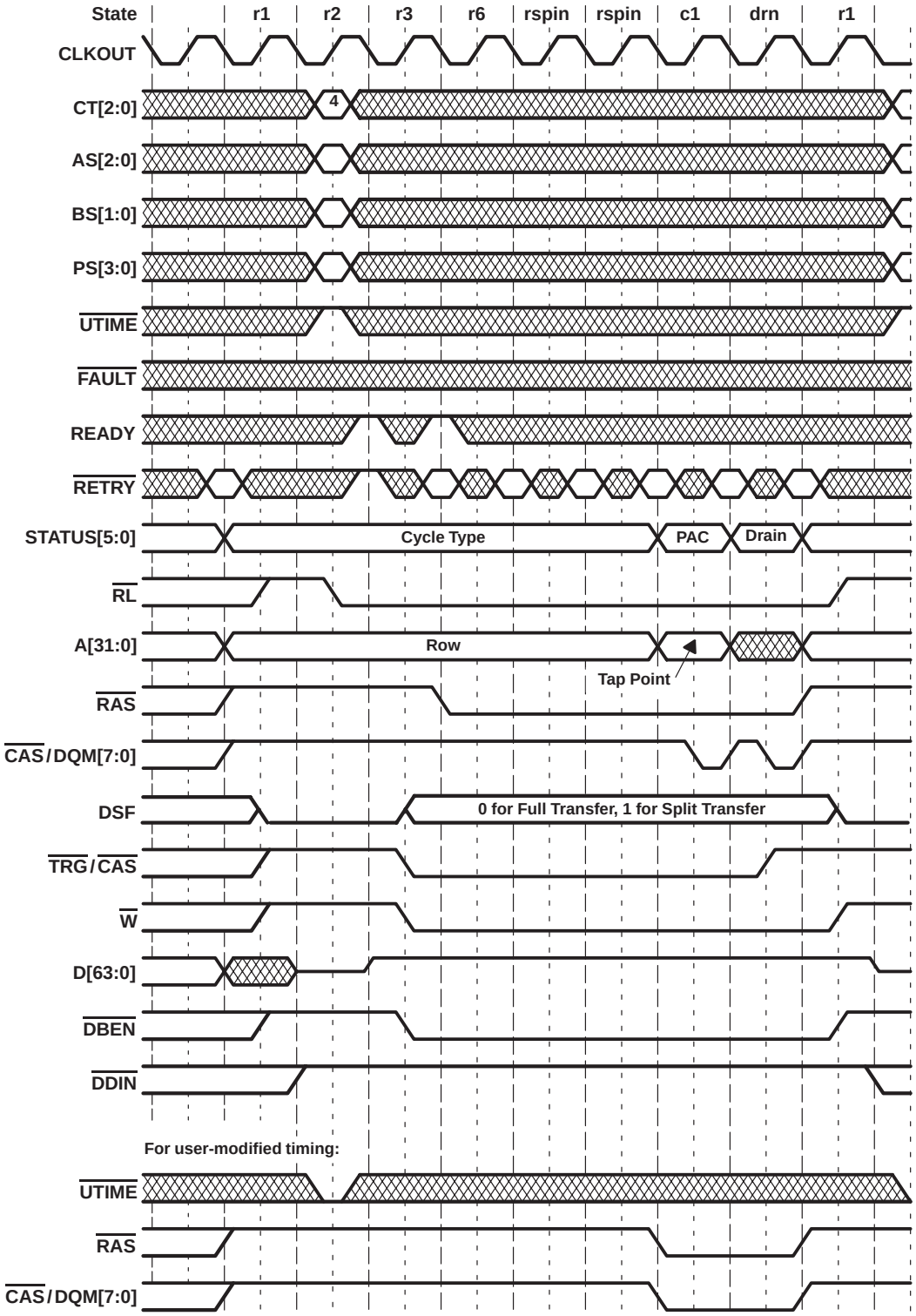


Figure 78. Pipelined 1-Cycle/Column Write-Transfer and Split-Register Write-Transfer-Cycle Timing

transfer cycles (continued)

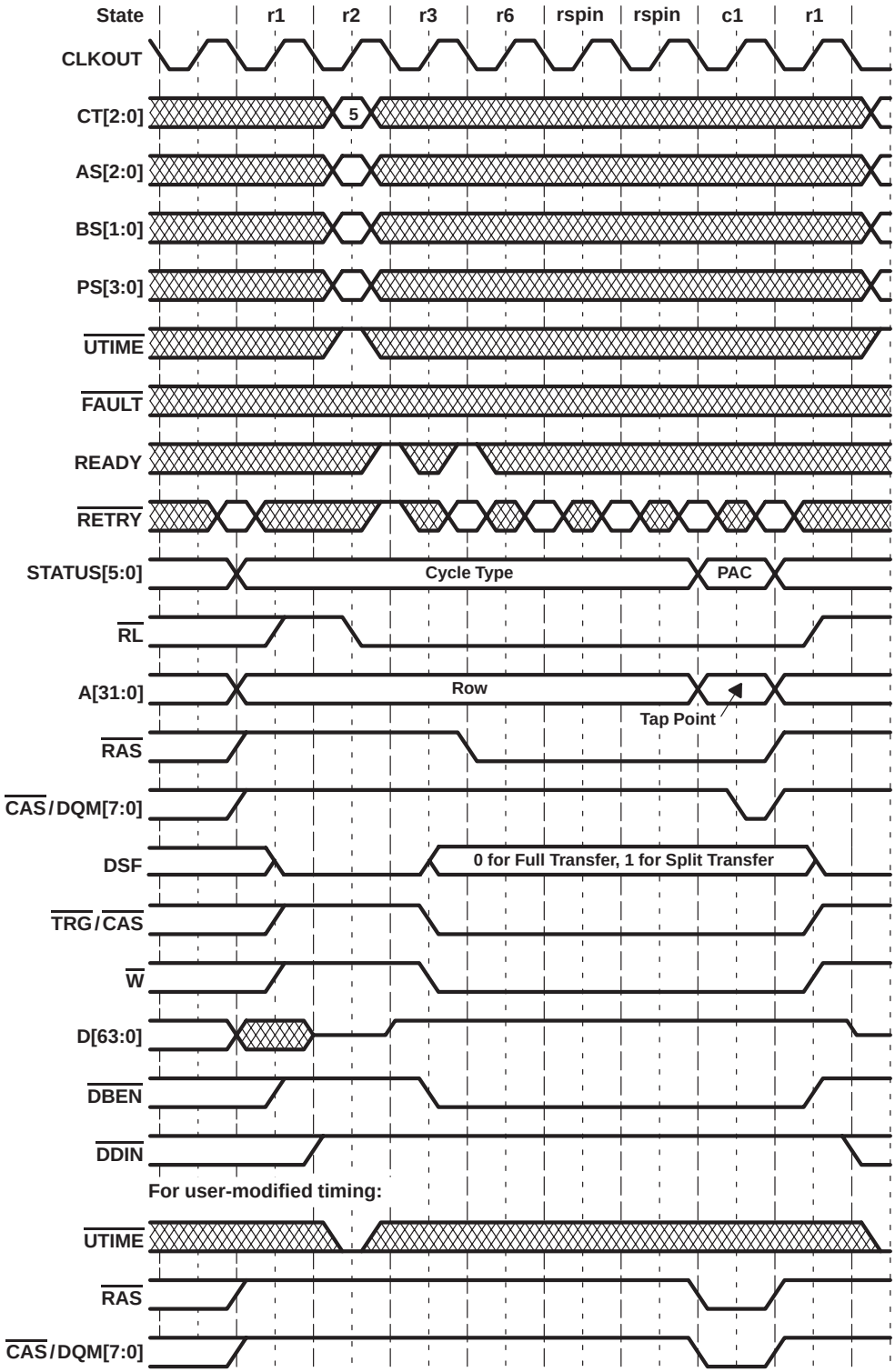


Figure 79. Nonpipelined 1-Cycle/Column Write-Transfer and Split-Register Write-Transfer-Cycle Timing

transfer cycles (continued)

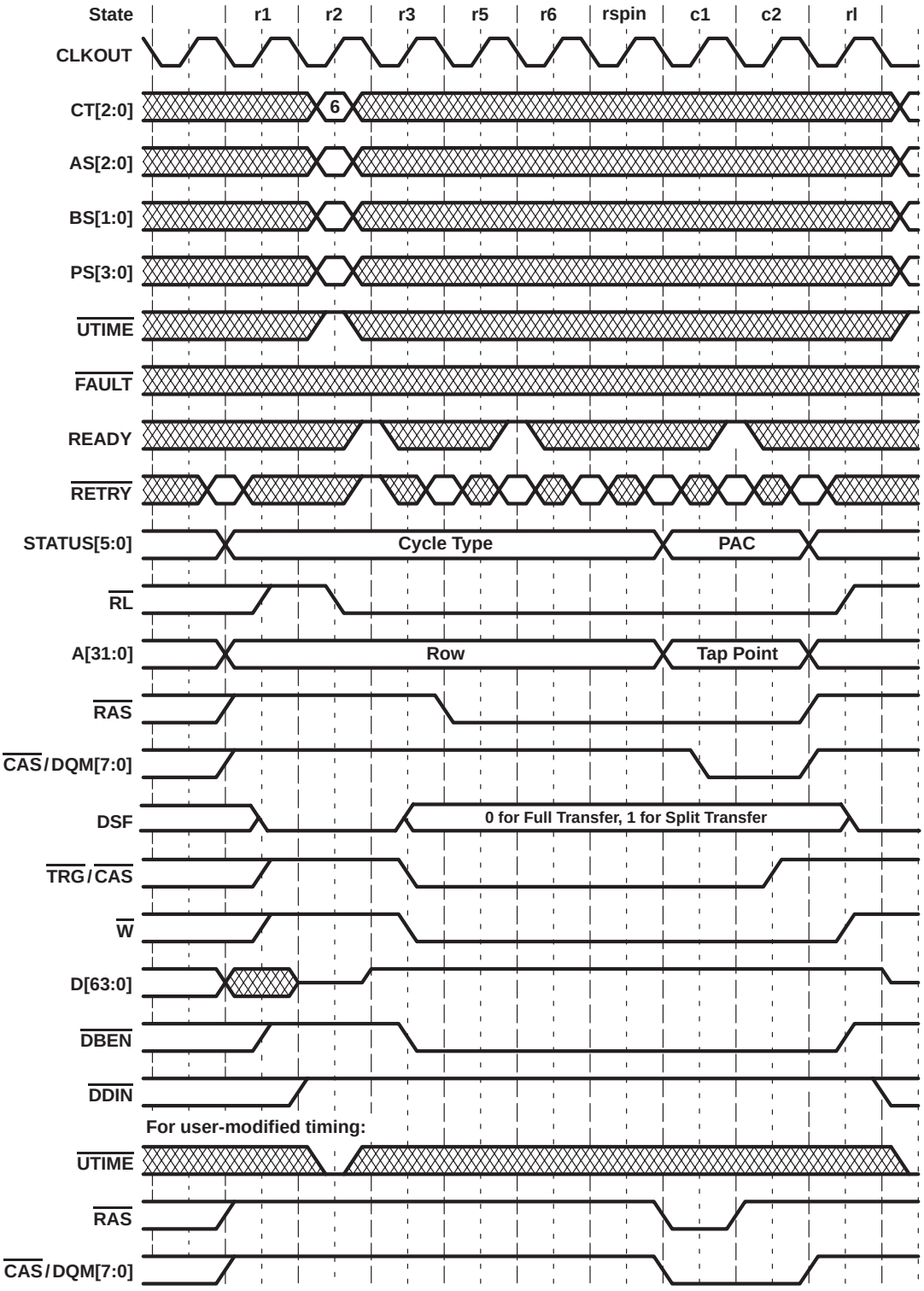


Figure 80. 2-Cycle/Column Write-Transfer and Split-Register Write-Transfer-Cycle Timing

transfer cycles (continued)

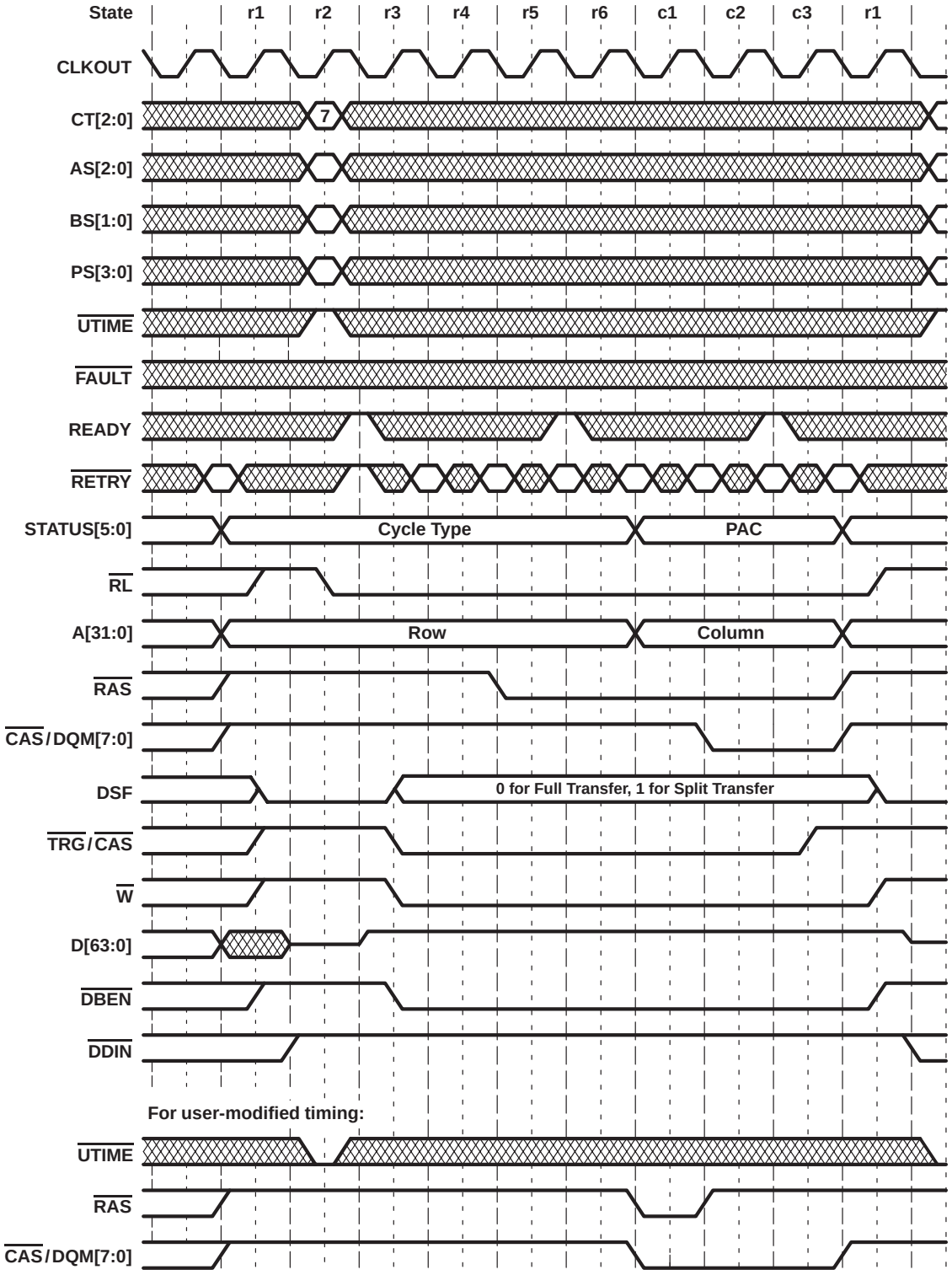


Figure 81. 3-Cycle/Column Write-Transfer and Split-Register Write-Transfer-Cycle Timing

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 ($\overline{\text{CAS}}$[7:0]) are active.
- $\overline{\text{TRG}}$, $\overline{\text{W}}$, and $\overline{\text{DBEN}}$ all remain inactive (high) because no data transfer occurs.
- DSF is active (high) at the fall of $\overline{\text{CAS}}$ and is driven inactive prior to the fall of $\overline{\text{RAS}}$.
- 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.

refresh cycles (continued)

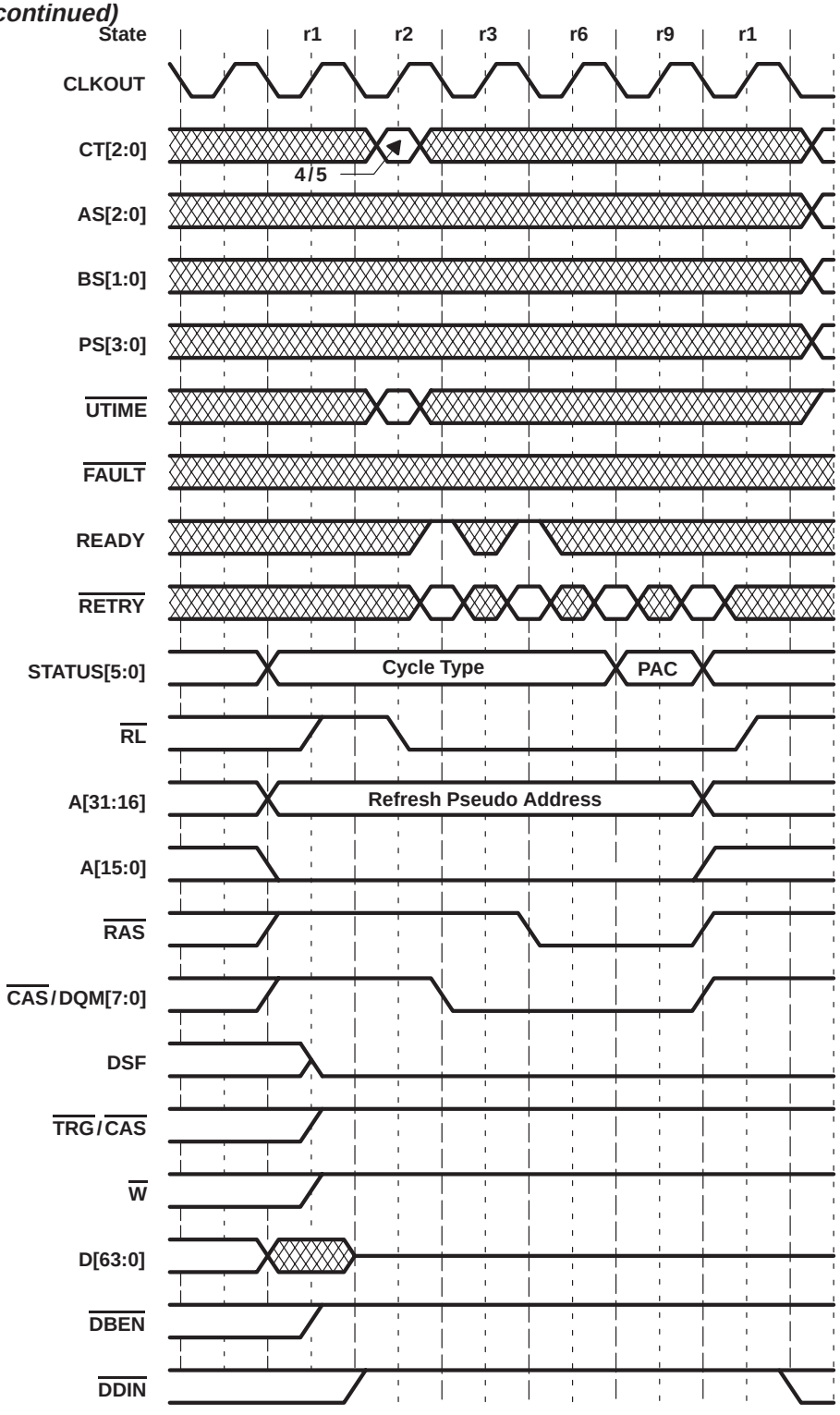


Figure 82. 1-Cycle/Column Refresh-Cycle Timing

refresh cycles (continued)

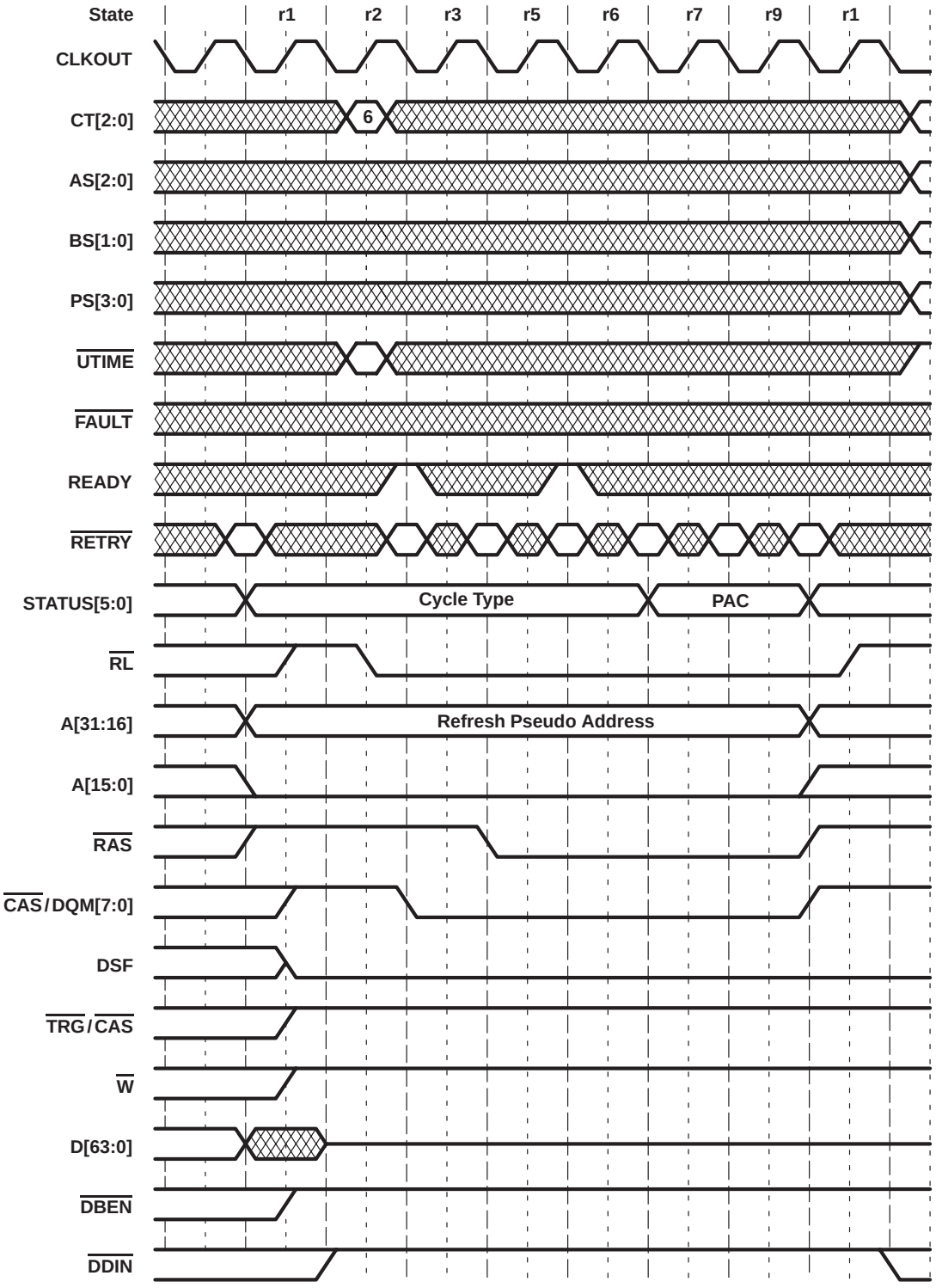


Figure 83. 2-Cycle/Column Refresh-Cycle Timing

refresh cycles (continued)

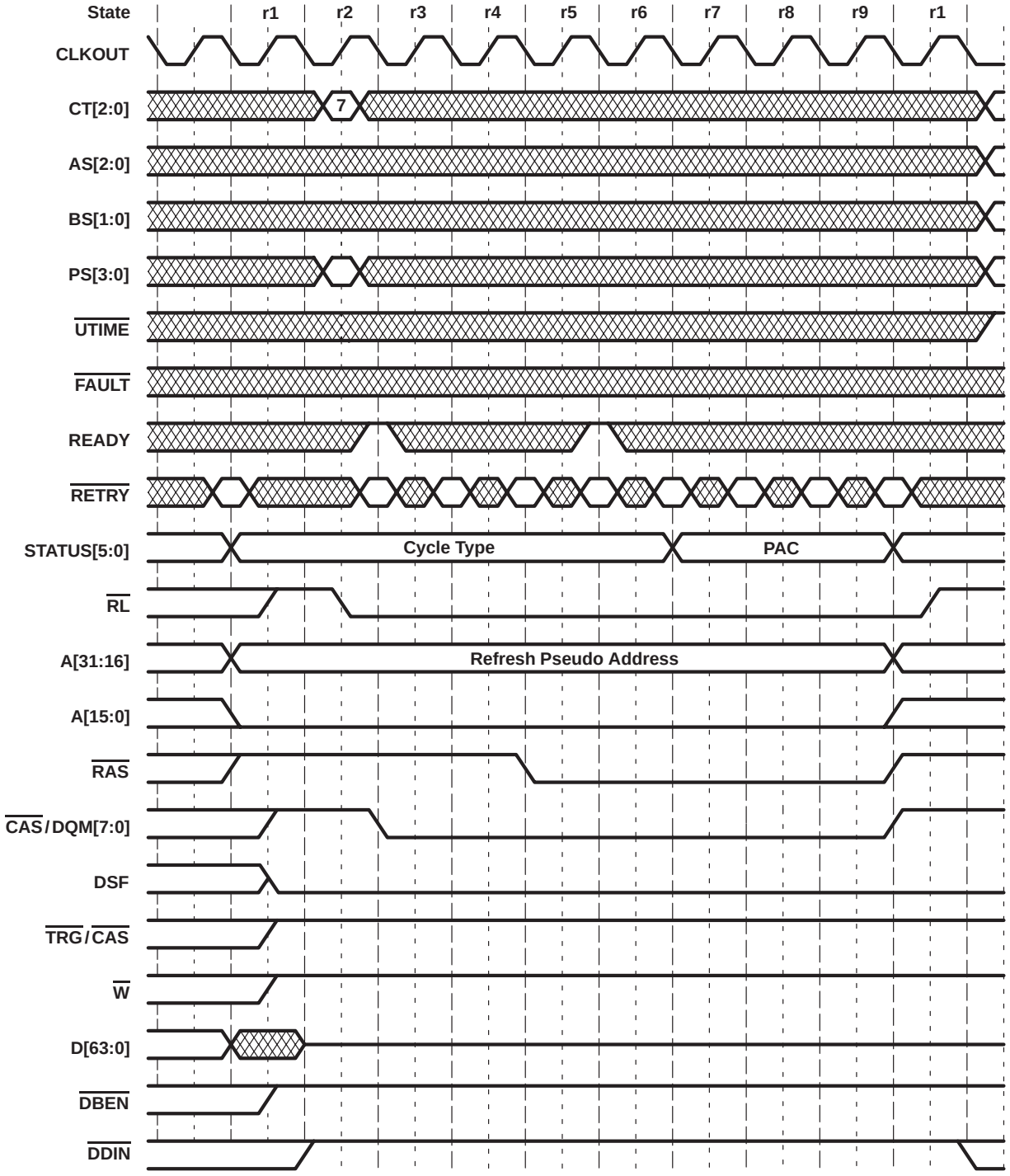


Figure 84. 3-Cycle/Column Refresh-Cycle Timing

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:

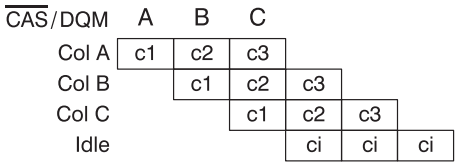
DCAB	Deactivate (precharge) all banks
ACTV	Activate the selected bank and select the row
READ	Input starting column address and start read operation
WRT	Input starting column address and start write operation
MRS	Set SDRAM mode register
REFR	Auto-refresh cycle with internal address
SRS	Set special register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

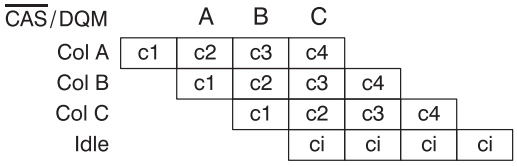
The SMJ320C80 supports CAS latencies of 2 or 3 cycles and burst lengths of 1 or 2. These are selected by the CT code input at the start of the access.

The column pipelines for SDRAM accesses are shown in Figure 85. 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.

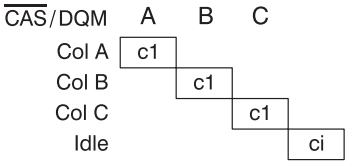
SDRAM type cycles (continued)



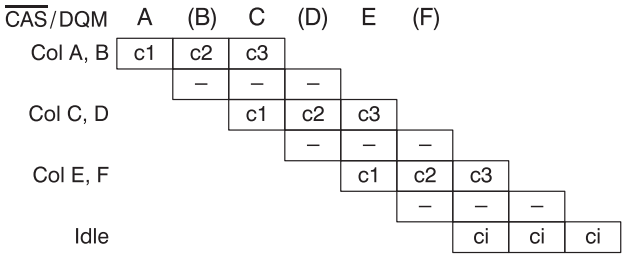
Burst-length 1, 2-cycle latency reads, read transfers, split-read transfers



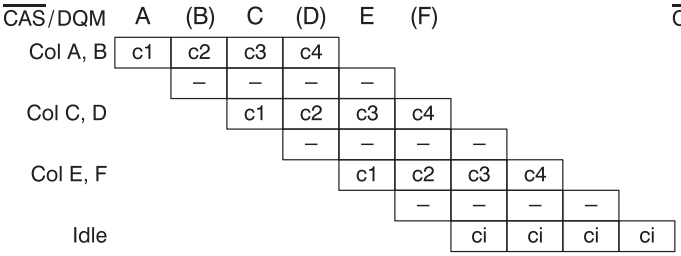
Burst-length 1, 3-cycle latency reads, read transfers, split-read transfers



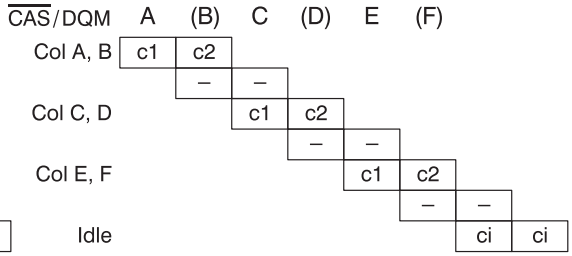
Burst-length 1 writes, block writes, SRSs, write transfers, split-write transfers



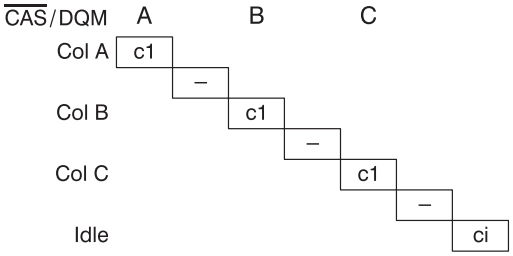
Burst-length 2, 2-cycle latency reads, read transfers, split-read transfers



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



Burst-length 2, 3-cycle latency writes



Burst-length 2, 3-cycle latency block-writes, write transfers, split-write transfers

Figure 85. SDRAM Column Pipelines

special SDRAM cycles

To initialize the SDRAM properly, the SMJ320C80 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.

special SDRAM cycles (continued)

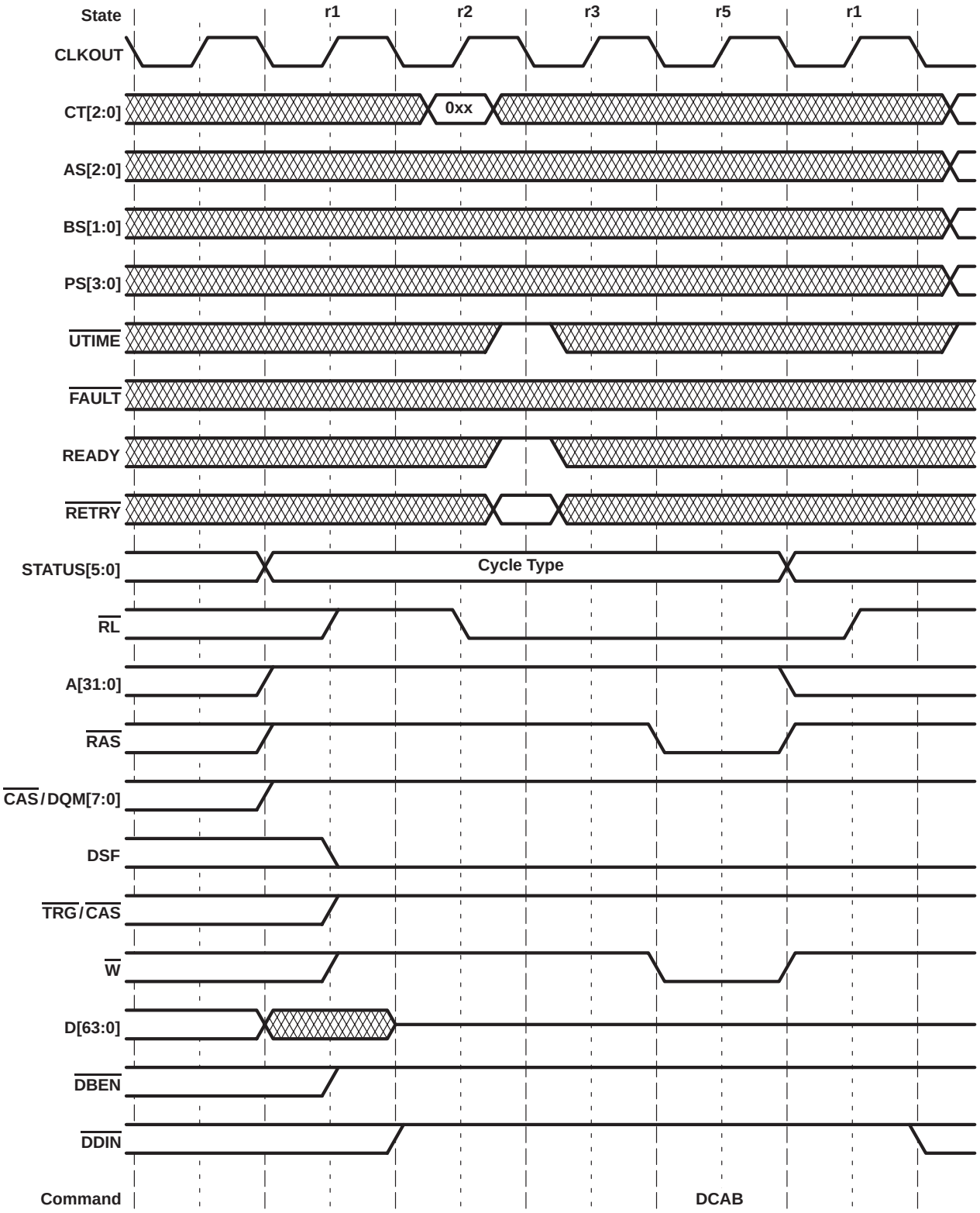


Figure 86. SDRAM Power-Up Deactivate Cycle Timing

special SDRAM cycles (continued)

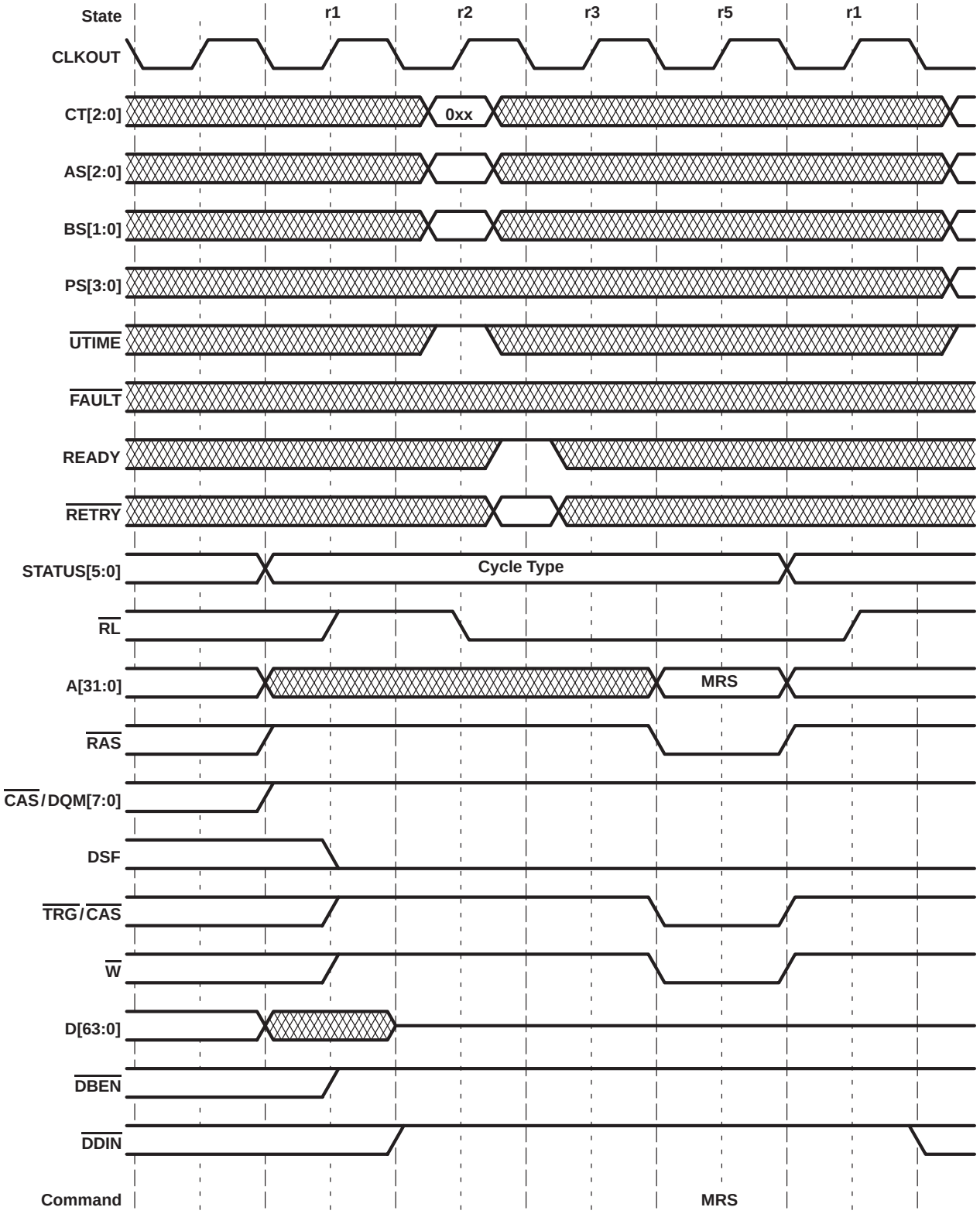


Figure 87. SDRAM Mode-Register-Set Cycle Timing

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 $\overline{\text{TRG}}$/$\overline{\text{CAS}}$ 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 $\overline{\text{CAS}}$/DQM strobes are activated two cycles before input data is latched. If the second column in a burst is not required, then $\overline{\text{CAS}}$/DQM is not activated. During peripheral device packet transfers, $\overline{\text{DBEN}}$ remains high.

SDRAM read cycles (continued)

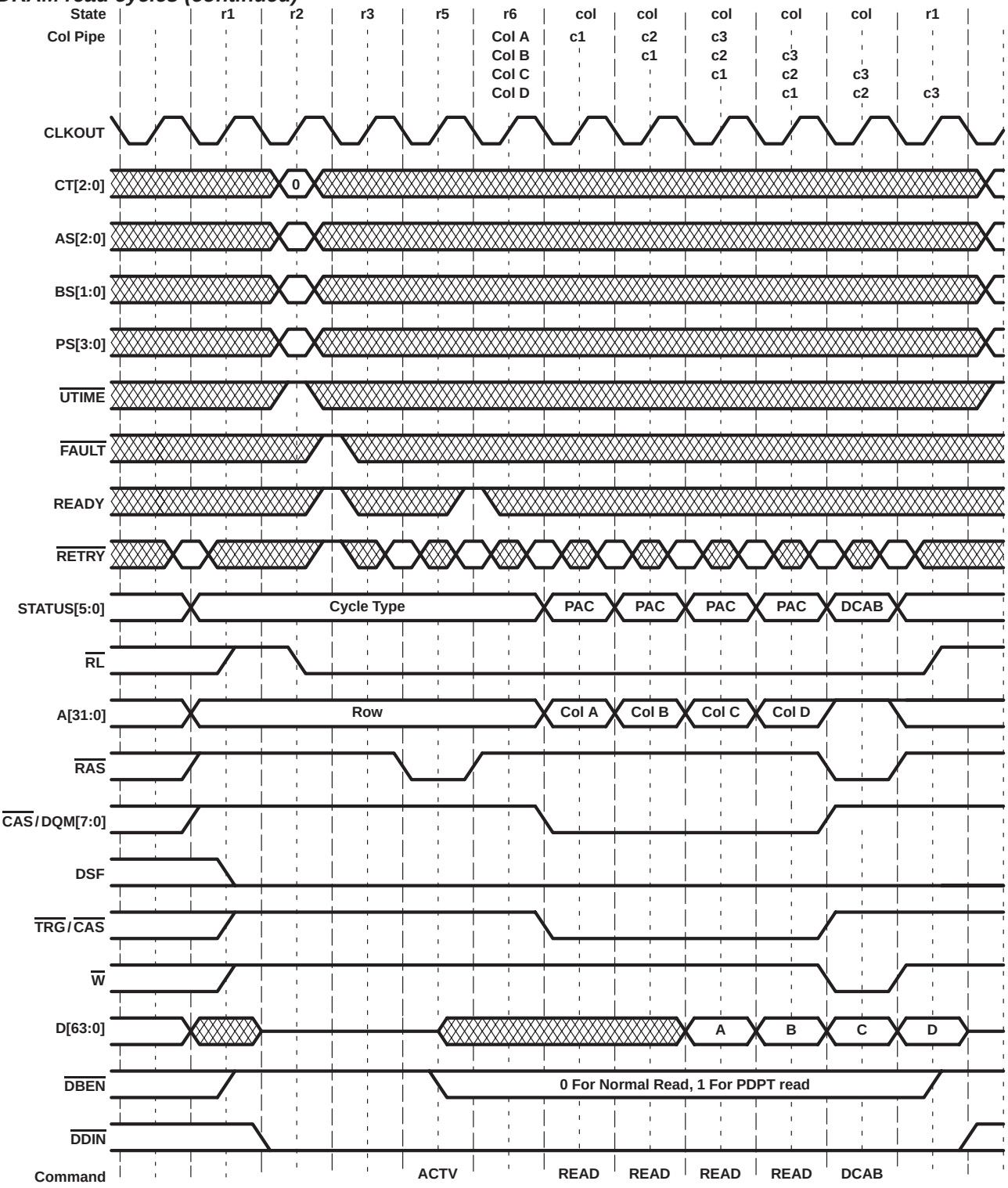


Figure 88. SDRAM Burst-Length 1, 2-Cycle Latency Read-Cycle Timing

SDRAM read cycles (continued)

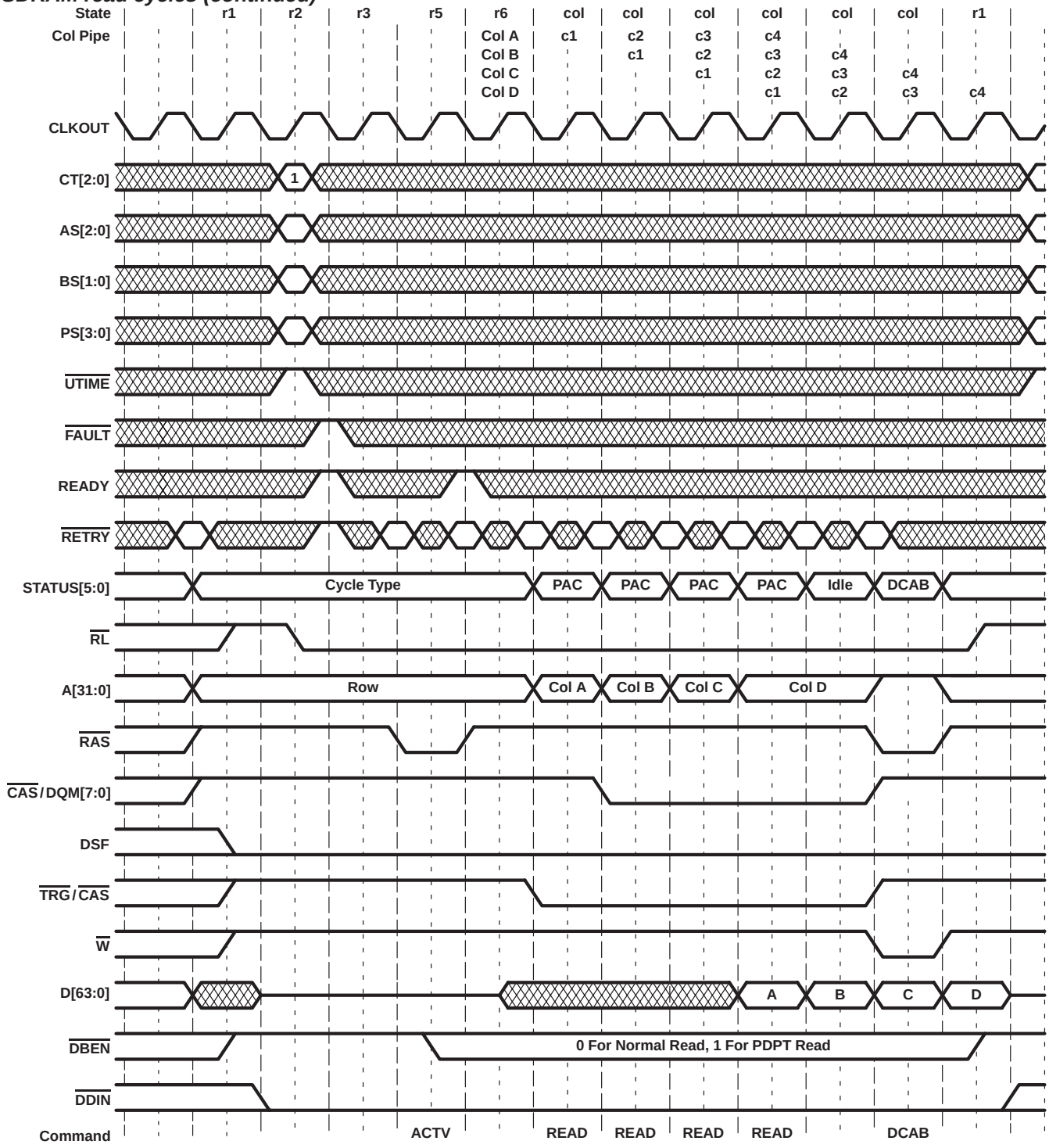


Figure 89. SDRAM Burst-Length 1, 3-Cycle Latency Read-Cycle Timing

SDRAM read cycles (continued)

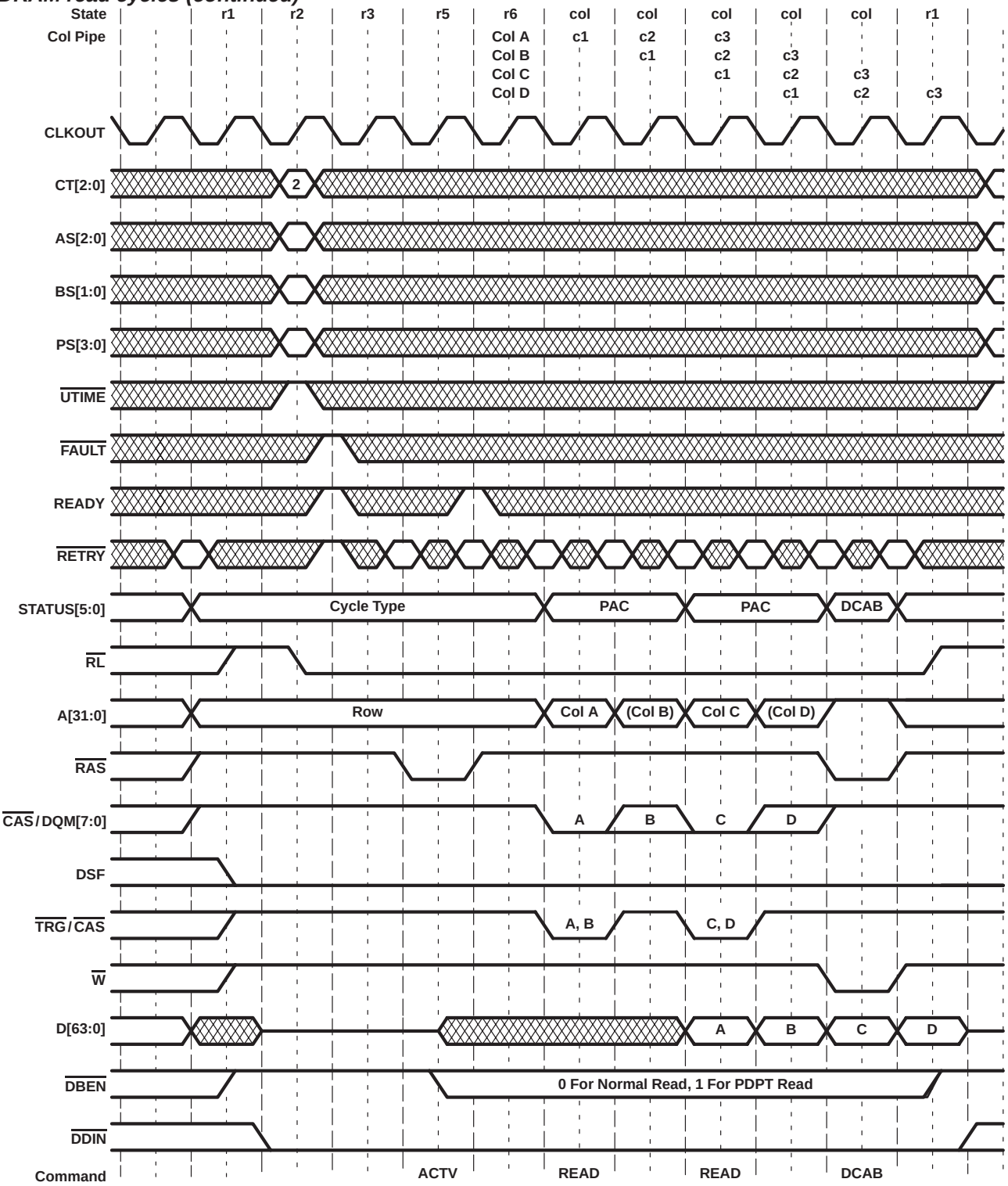


Figure 90. SDRAM Burst-Length 2, 2-Cycle Latency Read-Cycle Timing

SDRAM read cycles (continued)

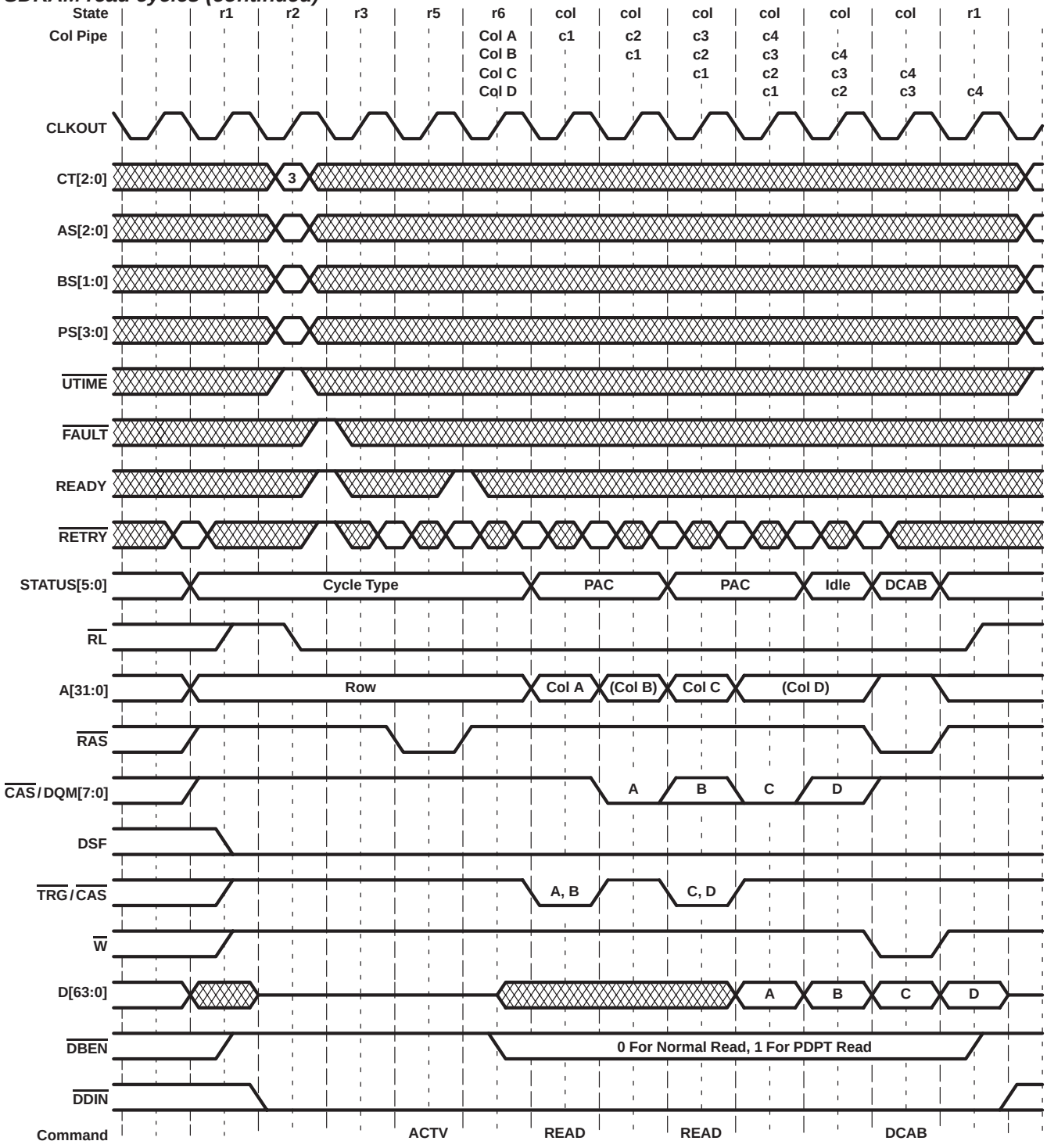


Figure 91. SDRAM Burst-Length 2, 3-Cycle Latency Read-Cycle Timing

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 $\overline{\text{TRG}}$/$\overline{\text{CAS}}$ and $\overline{\text{W}}$ 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 $\overline{\text{CAS}}$/DQM strobes low. During peripheral device packet transfers, $\overline{\text{DBEN}}$ remains high and D[63:0] are placed in the high-impedance state so that the peripheral can drive data into the memories.

SDRAM write cycles (continued)

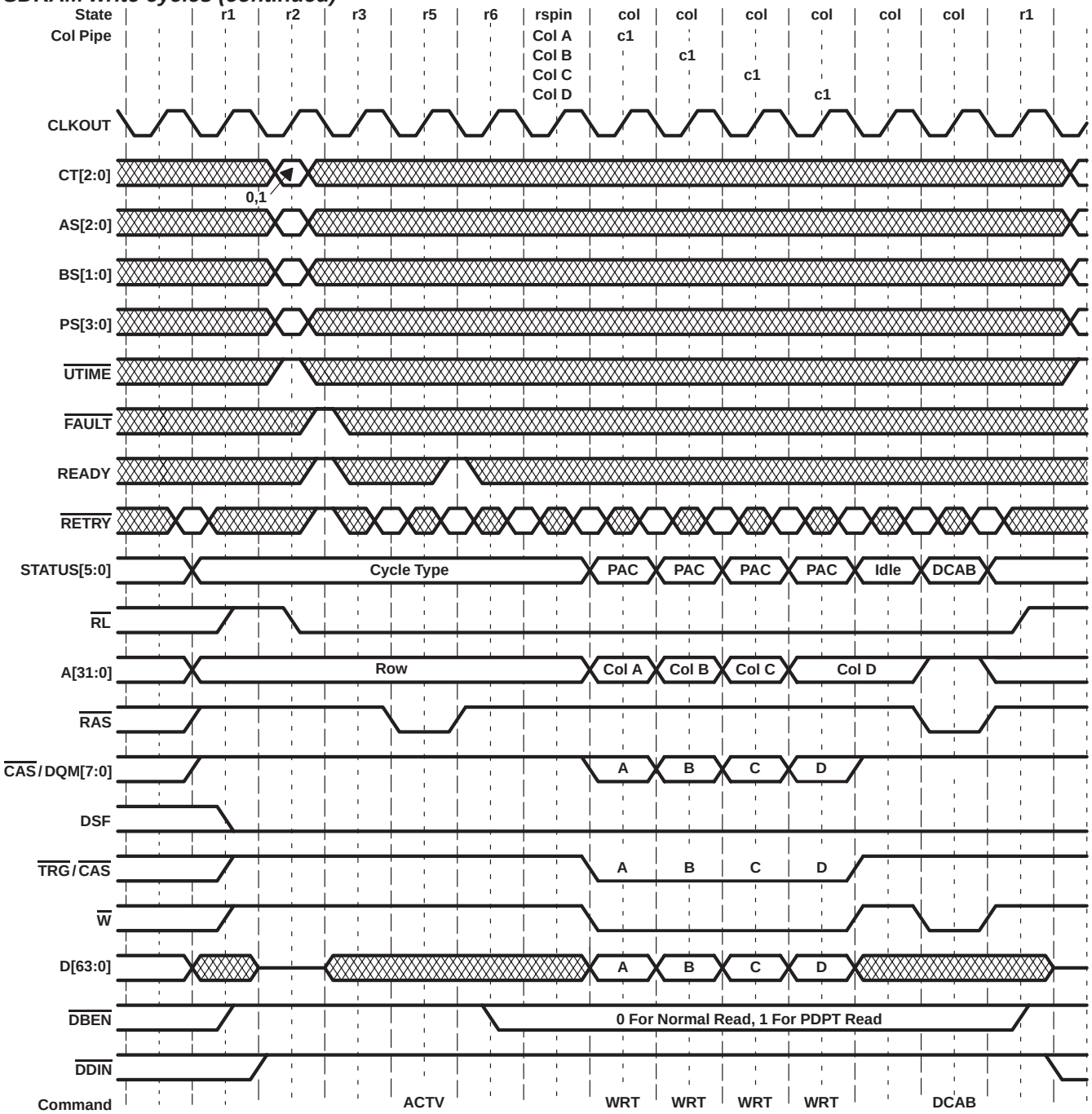


Figure 92. SDRAM Burst-Length 1 Write-Cycle Timing

SDRAM write cycles (continued)

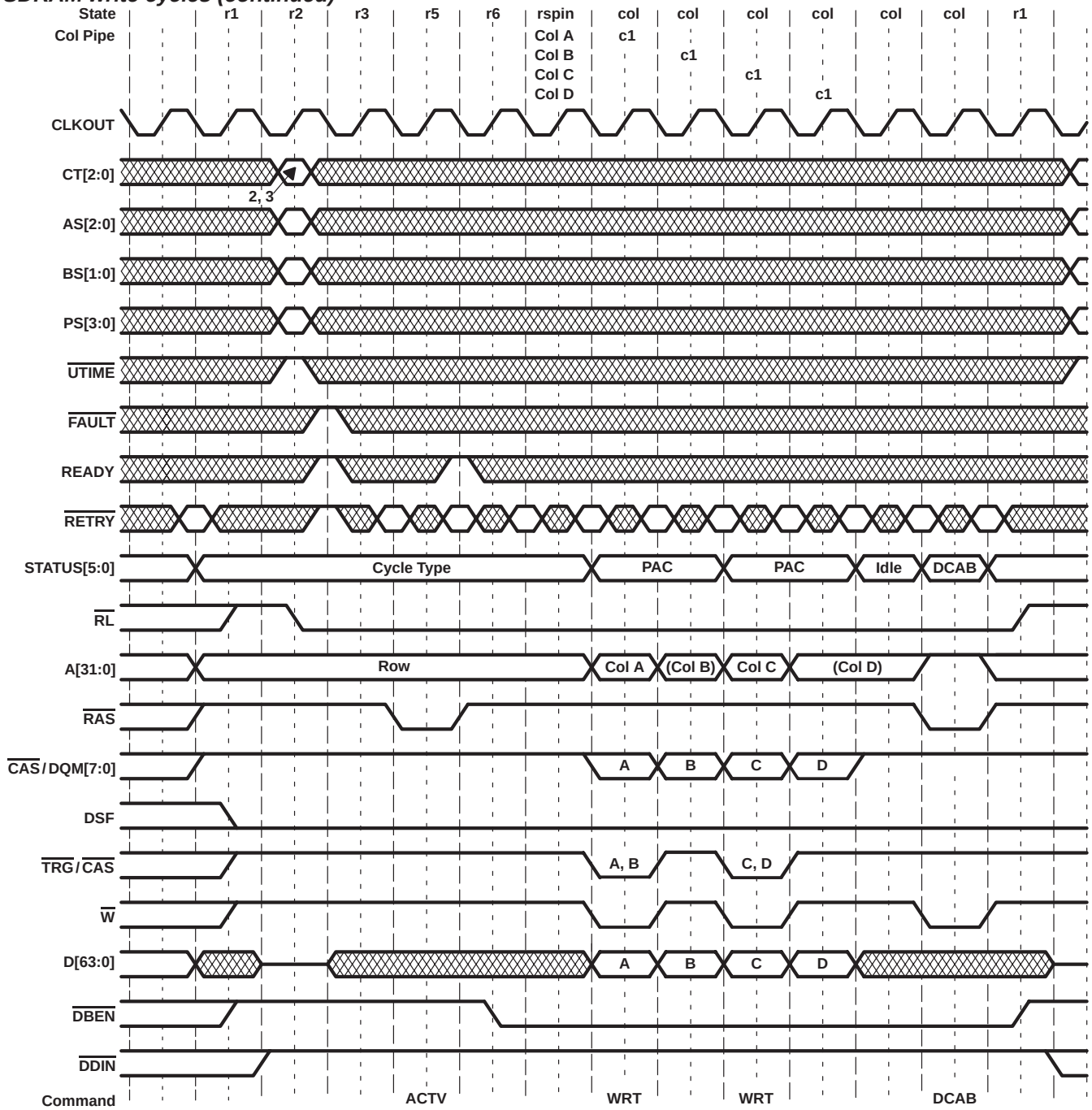


Figure 93. SDRAM Burst-Length 2 Write-Cycle Timing

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 94.

SDRAM Address Pin	BS	A8	A7	A6	A5	A4	A3	A2	A1	A0
SDRAM Function	0	0	0	LC	LM	LS	Stop Register			
SMJ320C80 Output Value	0	0	0	1	0	0	0	0	0	0

Figure 94. Special-Register-Set Value

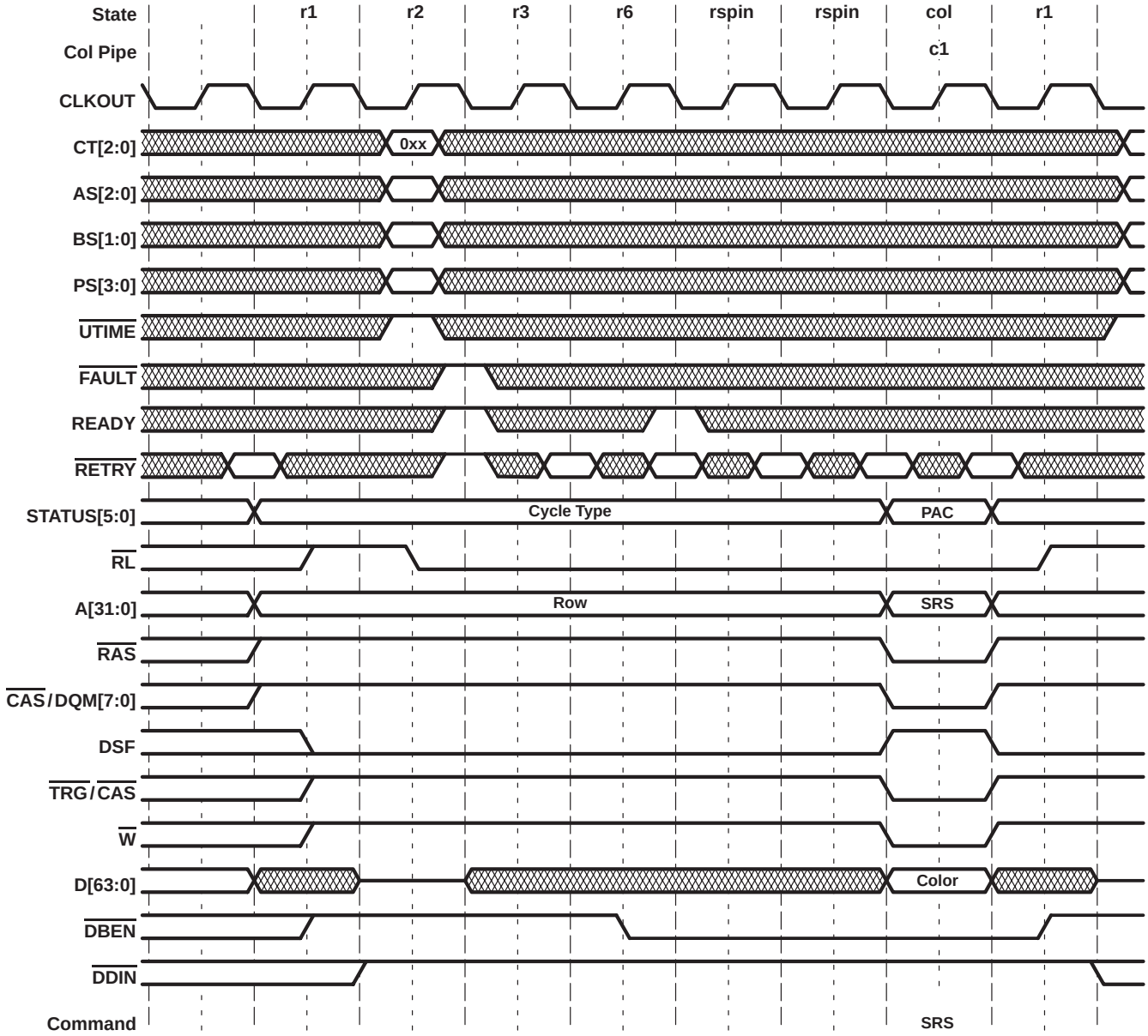


Figure 95. SDRAM SRS-Cycle Timing

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.

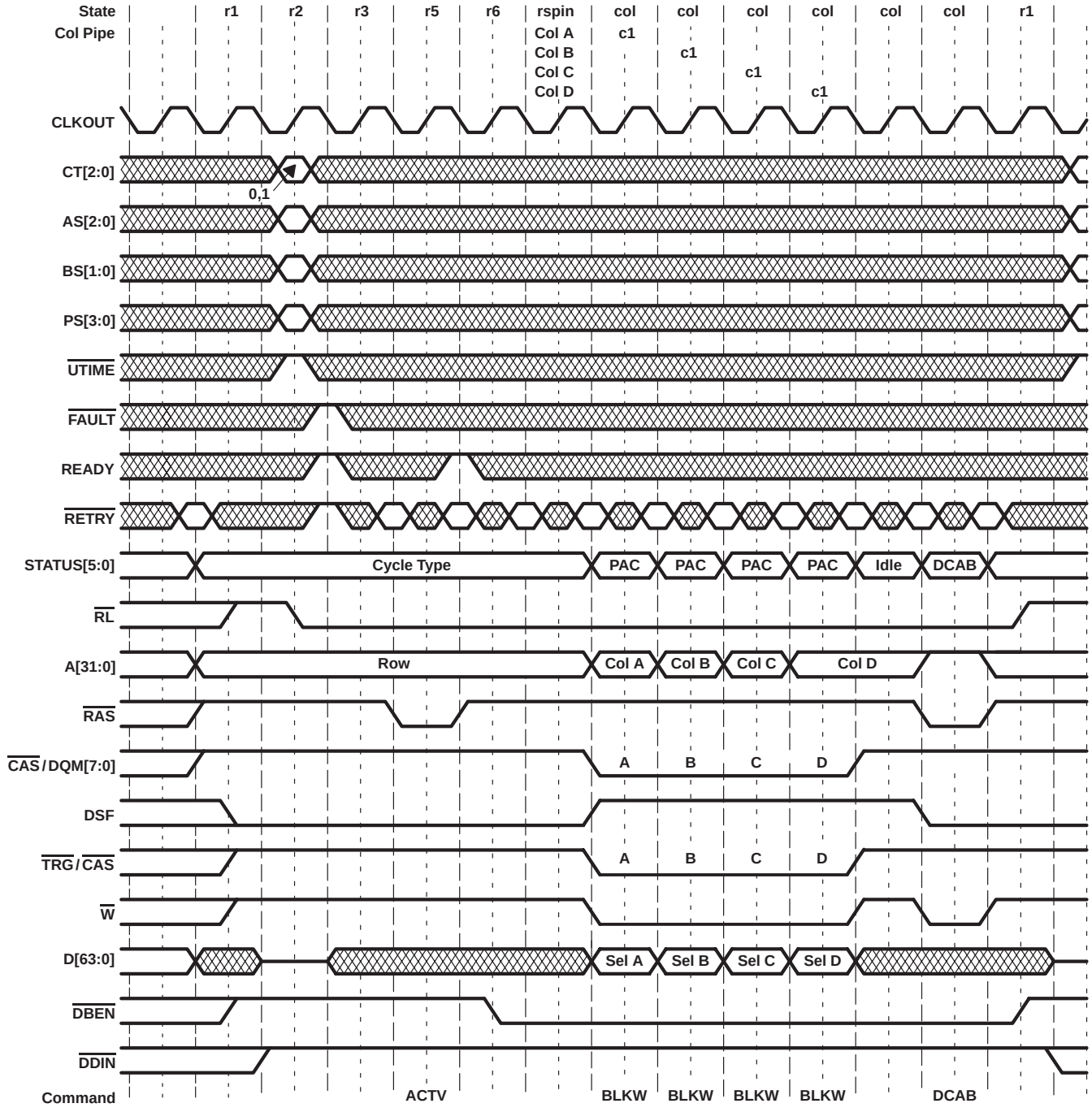


Figure 96. SDRAM Burst-Length 1 Block-Write Cycle Timing

SDRAM block-write cycles (continued)

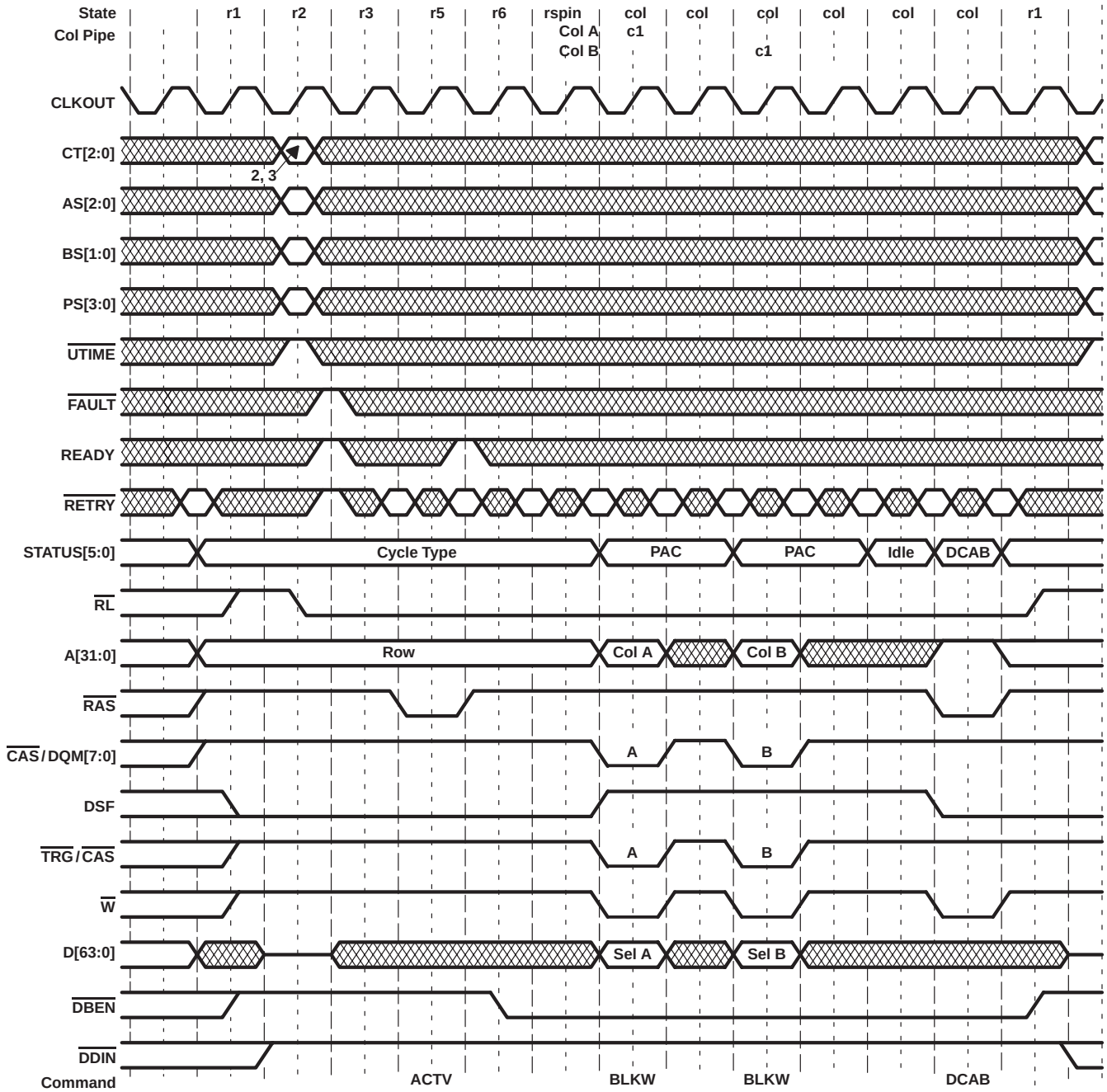


Figure 97. SDRAM Burst-Length 2 Block-Write Cycle Timing

SVRAM transfer cycles

The SVRAM read- and write-transfer cycles transfer data between the SVRAM memory-array and the serial register (SAM). The SMJ320C80 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.

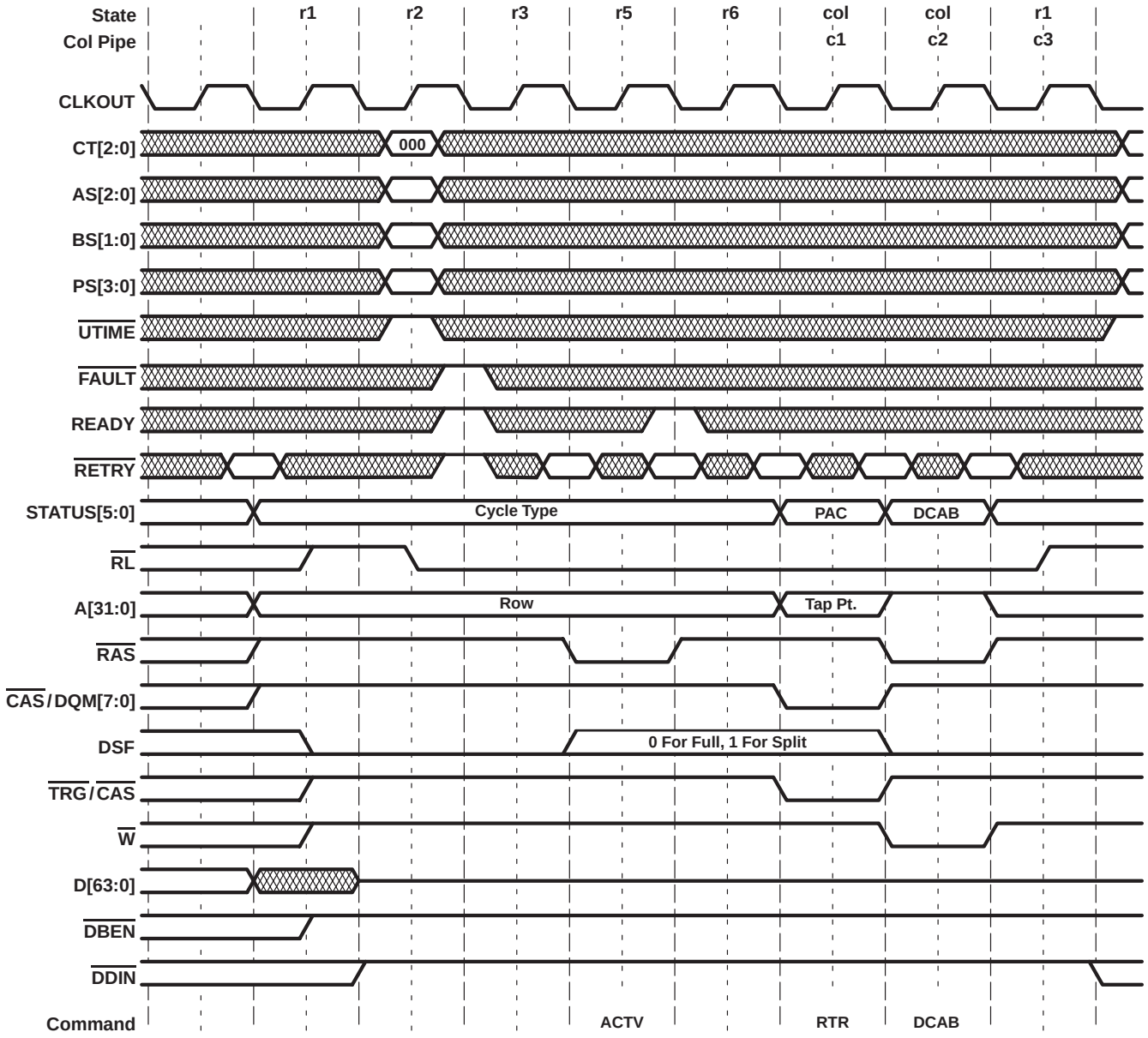


Figure 98. SVRAM Burst-Length 1, 2-Cycle Latency Read-Transfer Cycle Timing

SVRAM transfer cycles (continued)

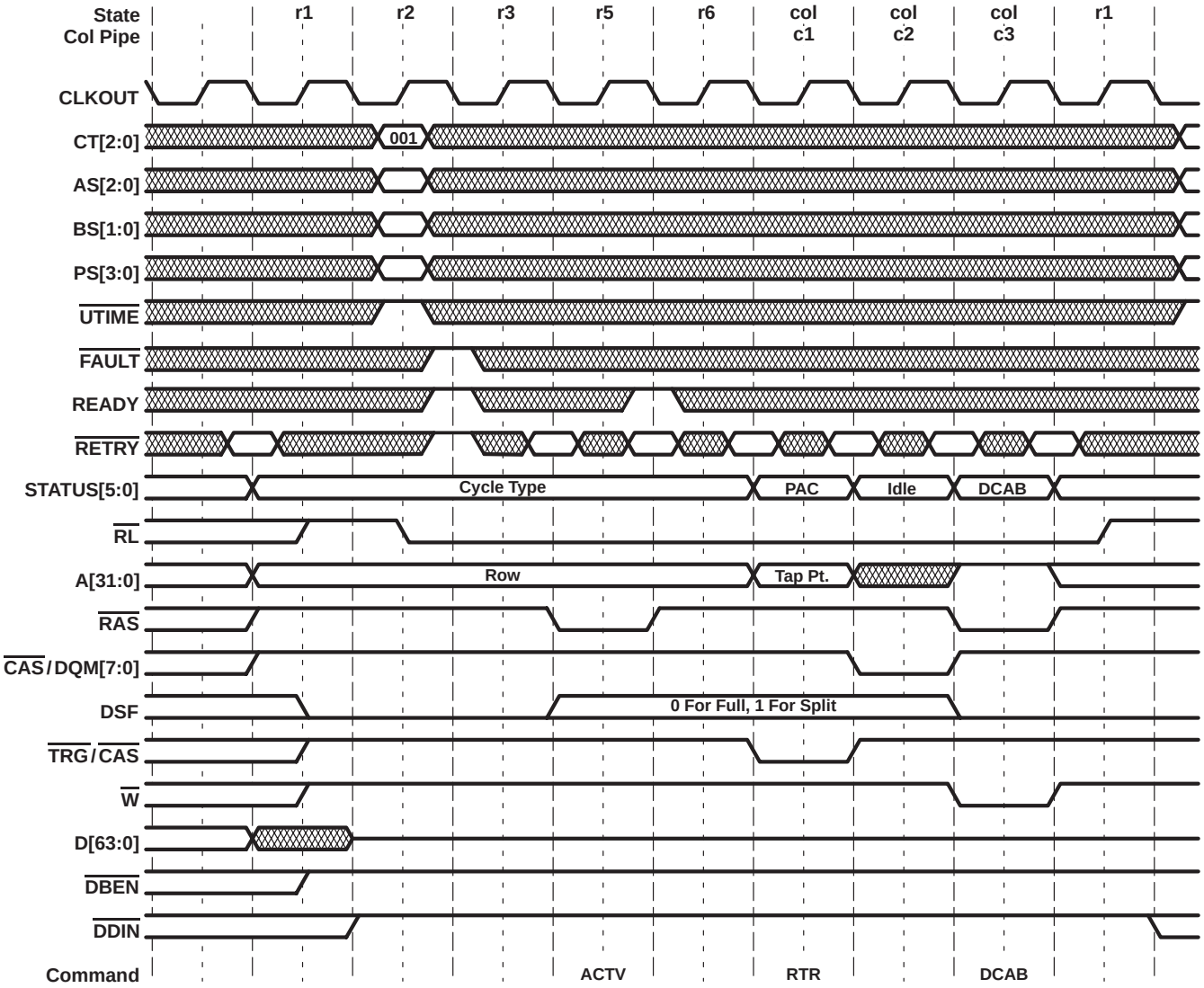


Figure 99. SVRAM Burst-Length 1, 3-Cycle Latency Read-Transfer Cycle Timing

SVRAM transfer cycles (continued)

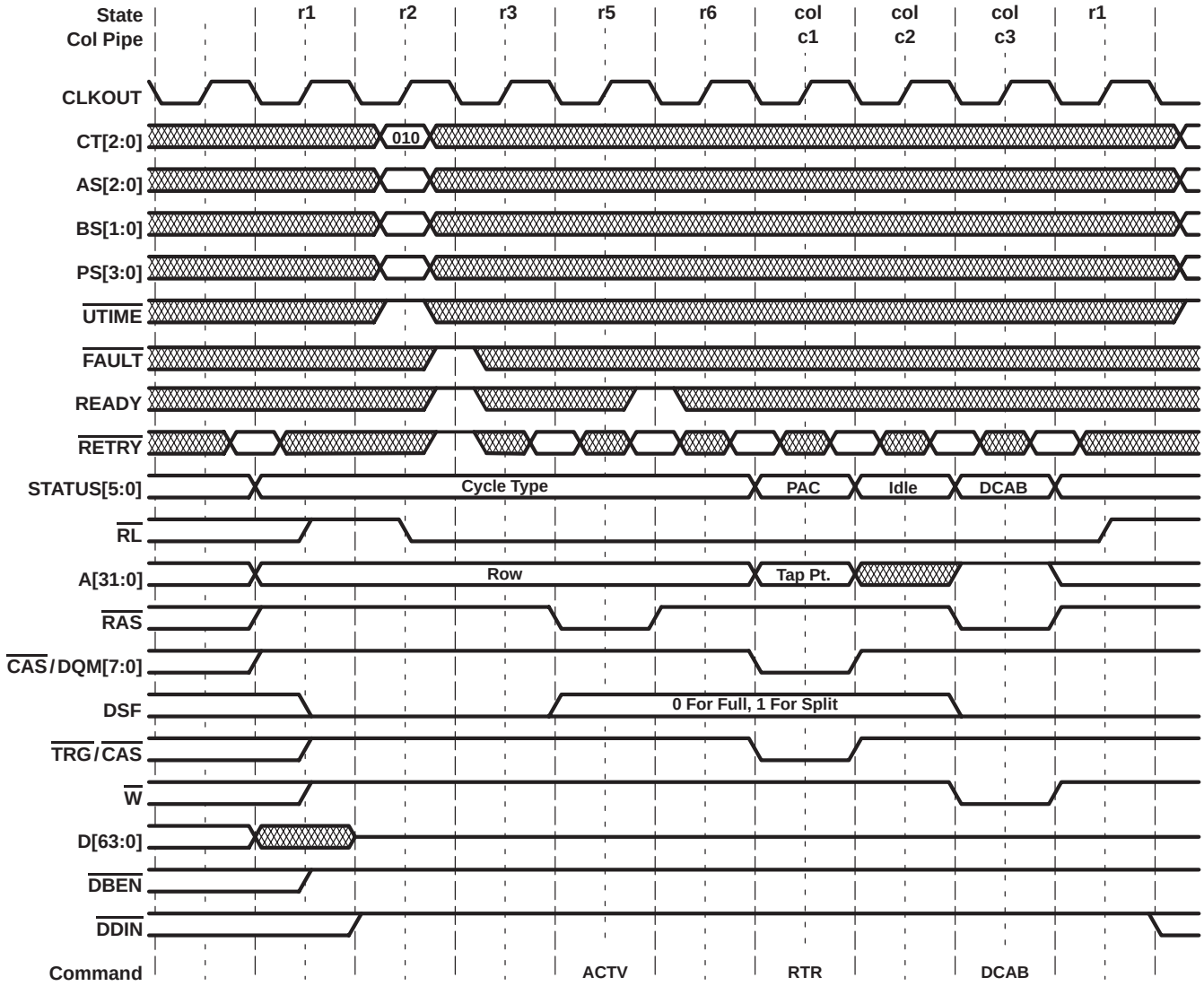


Figure 100. SVRAM Burst-Length 2, 2-Cycle Latency Read-Transfer Cycle Timing

SVRAM transfer cycles (continued)

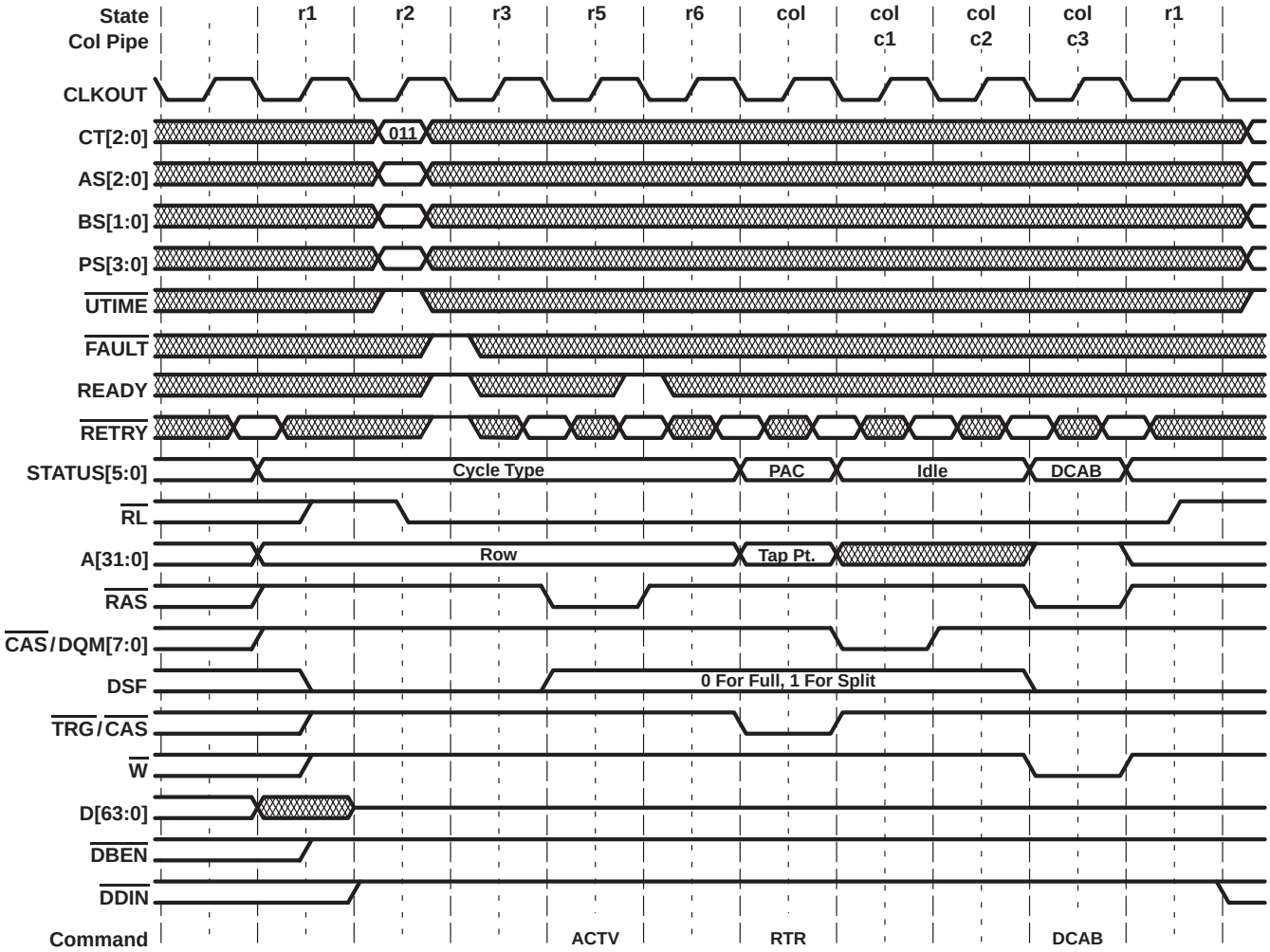


Figure 101. SVRAM Burst-Length 2, 3-Cycle Latency Read-Transfer Cycle Timing

SVRAM transfer cycles (continued)

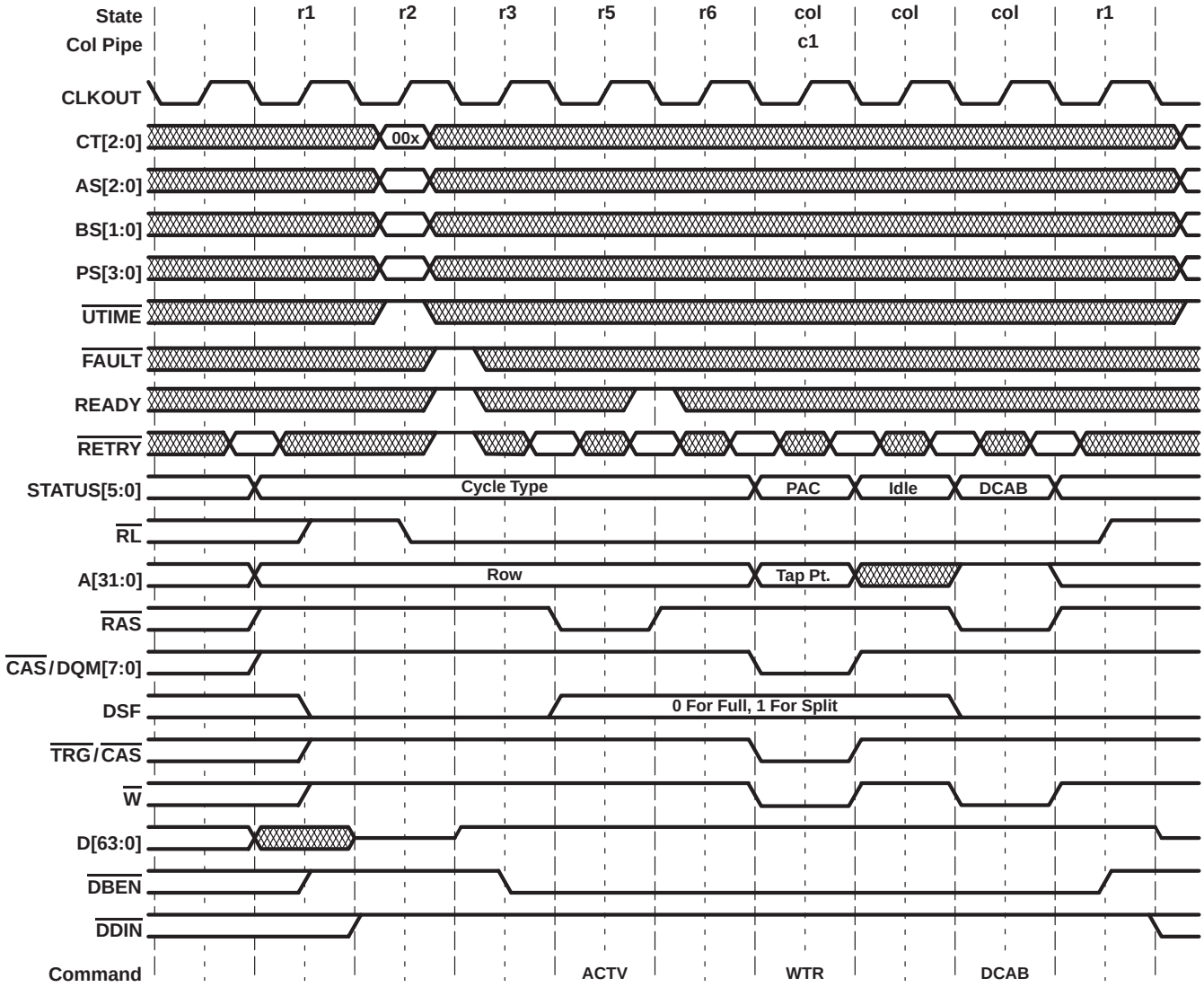


Figure 102. SVRAM Burst-Length 1, Write-Transfer Cycle Timing

SVRAM transfer cycles (continued)

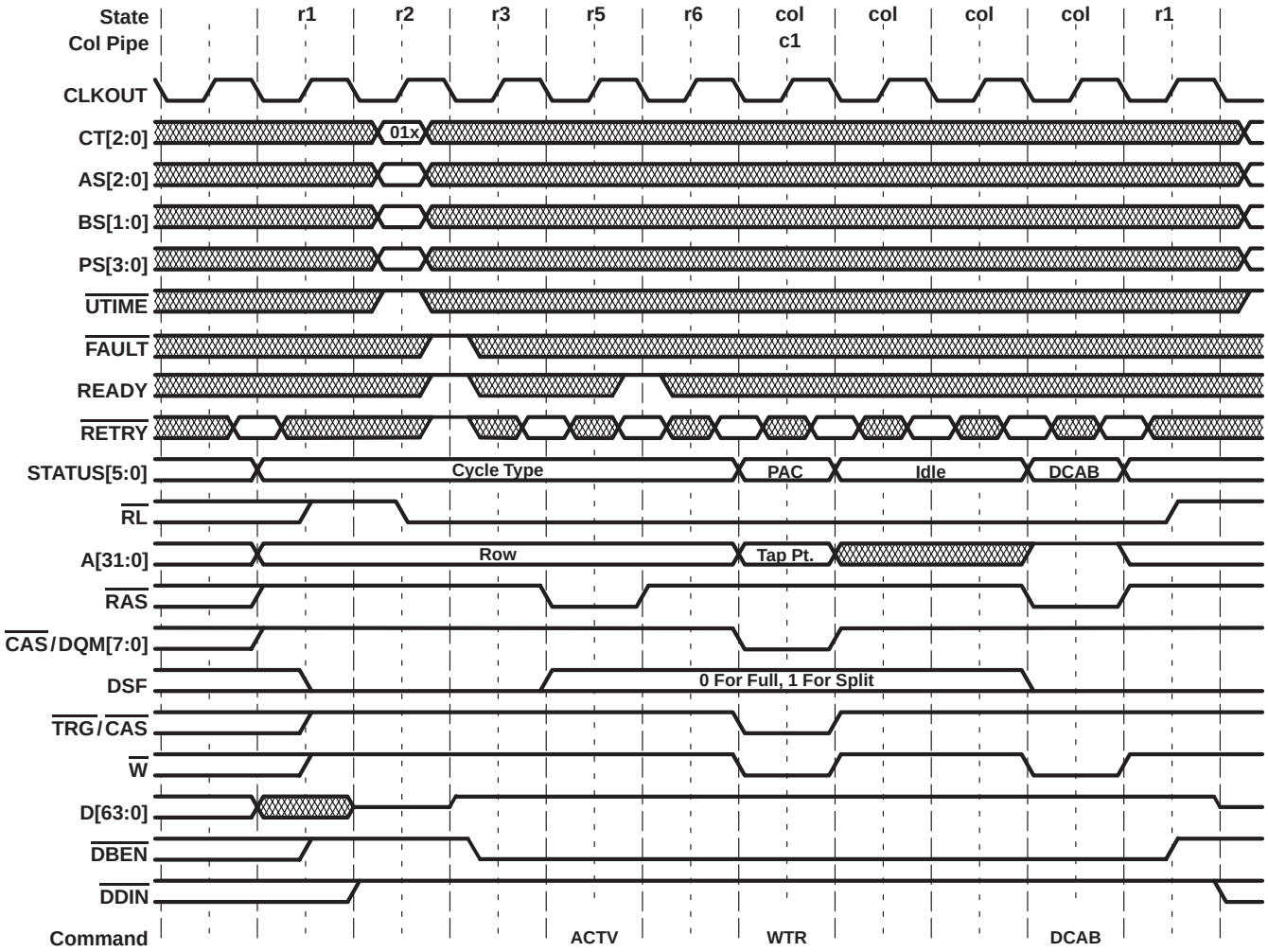


Figure 103. SVRAM Burst-Length 2, Write-Transfer Cycle Timing

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 $\overline{RAS}$ and $\overline{TRG}/\overline{CAS}$ 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.

SDRAM refresh cycle (continued)

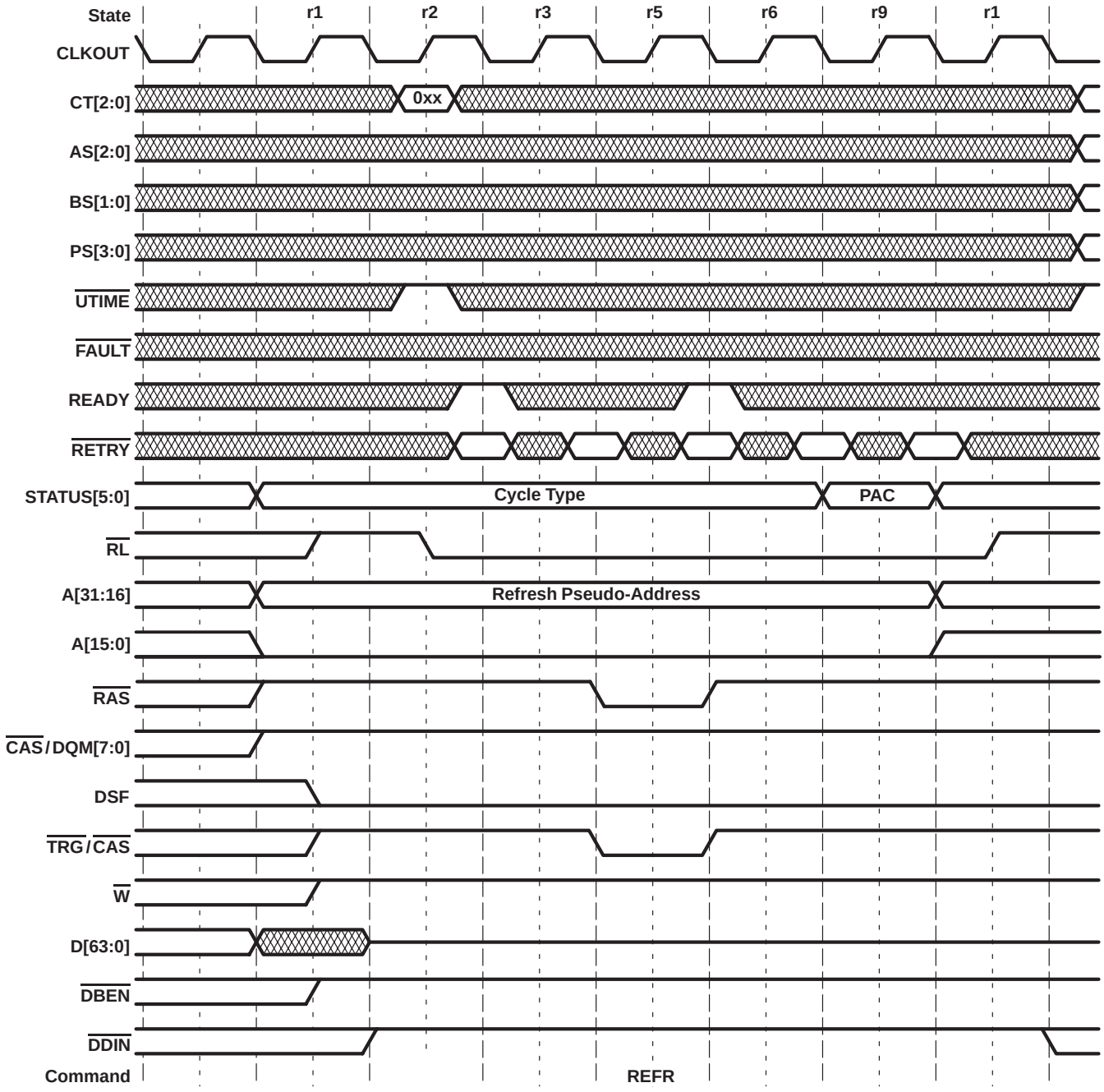


Figure 104. SDRAM Refresh Cycle Timing

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 $\overline{\text{HREQ}}$ 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 $\overline{\text{HACK}}$ 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.

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 SMJ320C80 is reset when the $\overline{\text{RESET}}$ input is driven low. The ’C80 outputs immediately go into a high-impedance state with the exception of CLKOUT, $\overline{\text{HACK}}$, and REQ[1:0]. While $\overline{\text{RESET}}$ is low, all internal registers are set to their default values and internal logic is reset.

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

Once $\overline{\text{RESET}}$ 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 $\overline{\text{EINT3}}$ 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
Ambient air temperature, T_A . . . . . – 55°C to 125°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	0	0	V
I_{OH}	High-level output current			– 400	µA
I_{OL}	Low-level output current			2	mA
T_C	Case temperature			125	°C
T_A	Ambient-air temperature	– 55			°C

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

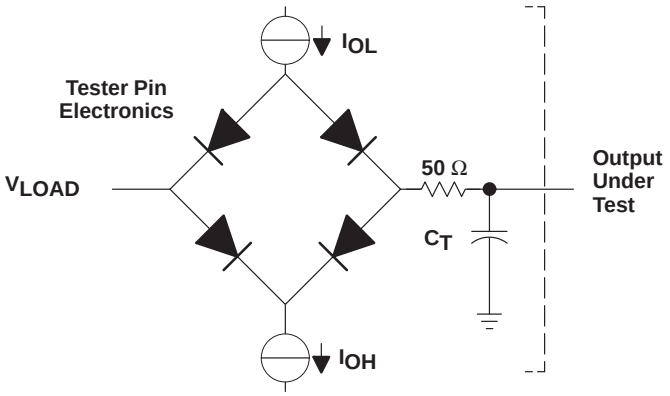
electrical characteristics over recommended range of supply voltage and specified temperature (unless otherwise noted)

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	1.85			V
V_{OL}	Low-level output voltage	V_{DD} = MAX				0.9	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), TRST	V_I = V_{SS} to V_{DD}				±20	µA
I_{DD}	Supply current (see Note 3)	V_{DD} = MAX,	50 MHz		1§	2.5	A
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, T_A = 25°C
¶ Typical steady-state V_{OH} will not exceed V_{DD}

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} = 2.2 V
C_T = 60 pF typical load circuit capacitance

Figure 105. Test Load Circuit

signal transition levels

TTL-output levels are driven to a minimum logic-high level of 1.85 V and to a maximum logic-low level of 0.9 V. Figure 106 shows the TTL-level outputs.

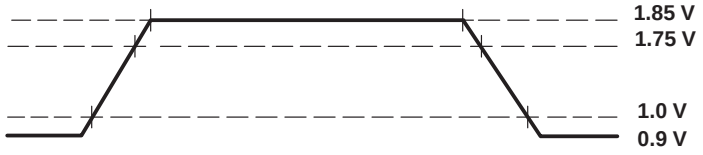


Figure 106. TTL-Level Outputs

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 1.75 V, and the level at which the output is said to be low is 1.0 V.
- For a low-to-high transition, the level at which the output is said to be no longer low is 1.0 V, and the level at which the output is said to be high is 1.75 V.

Figure 107 shows the TTL-level inputs.

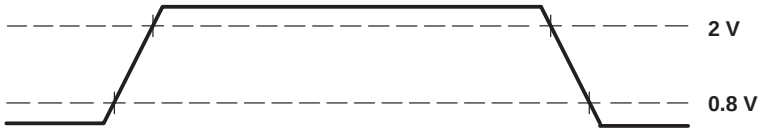


Figure 107. TTL-Level Inputs

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

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 108)

NO			MIN	MAX	UNIT
1	$t_{c(CKI)}$	Period of CLKIN (t_H)	10		ns
2	$t_{w(CKIH)}$	Pulse duration of CLKIN high	4.2		ns
3	$t_{w(CKIL)}$	Pulse duration of CLKIN low	4.2		ns
4	$t_{t(CKI)}$	Transition time of CLKIN		1.5*	ns

* This parameter is not production tested.

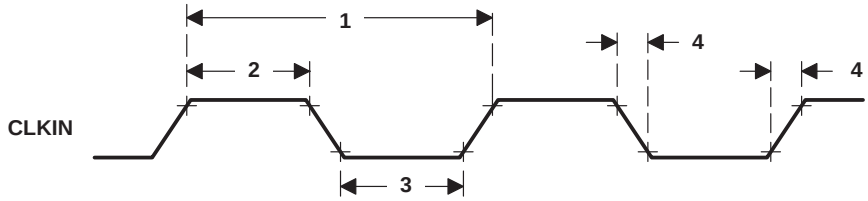


Figure 108. CLKIN Timing

local-bus switching characteristics over full operating range: CLKOUT† (see Figure 109)

NO	PARAMETER		MIN	MAX	UNIT
5	$t_{c(CKO)}$	Period of CLKOUT	$2t_{c(CKI)}$‡*		ns
6	$t_{w(CKOH)}$	Pulse duration of CLKOUT high	$t_H-4.5$		ns
7	$t_{w(CKOL)}$	Pulse duration of CLKOUT low	$t_H-4.5$		ns
8	$t_{t(CKO)}$	Transition time of CLKOUT		2.5*	ns

† The CLKOUT output has twice the period of CLKIN. No propagation delay or phase relationship to CLKIN is ensured. Each state of a memory access begins on the falling edge of CLKOUT.
‡ This parameter can also be specified as $2t_H$.
* This parameter is not production tested.

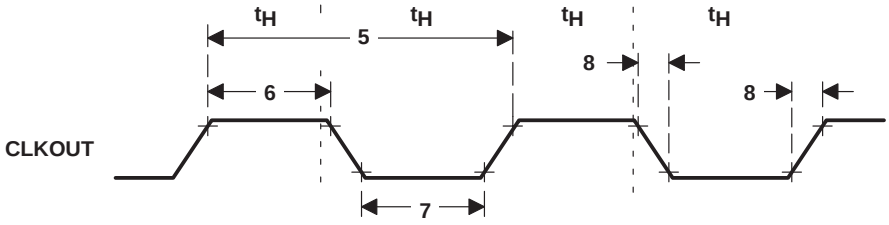


Figure 109. CLKOUT Timing

device reset timing requirements (see Figure 110)

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	$t_{su(HRQL-RSTH)}$	Setup time of $\overline{HREQ}$ low to $\overline{RESET}$ high to configure self-bootstrap mode		$4t_h$		ns
11	$t_{h(RSTH-HRQL)}$	Hold time, $\overline{HREQ}$ low after $\overline{RESET}$ high to configure self-bootstrap mode		0		ns
12	$t_{su(UTML-RSTH)}$	Setup time of $\overline{UTIME}$ low to $\overline{RESET}$ high to configure big-endian operation		$4t_h$		ns
13	$t_{h(RSTH-UTML)}$	Hold time, $\overline{UTIME}$ low after $\overline{RESET}$ high to configure big-endian operation		0		ns

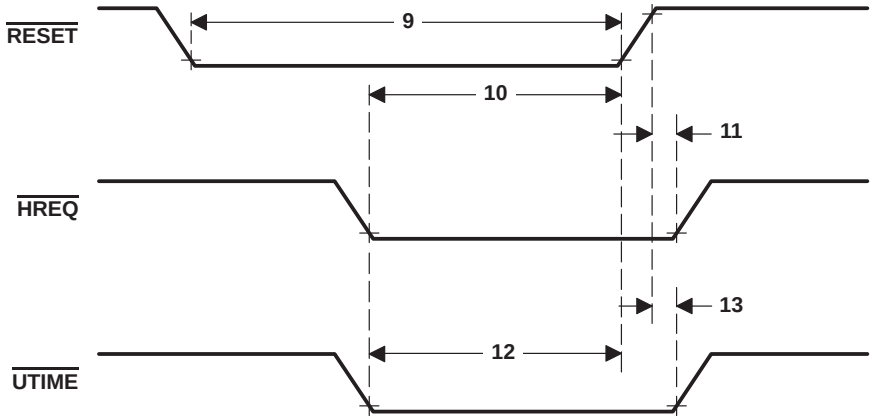


Figure 110. Device-Reset Timing

local bus timing requirements: cycle configuration inputs (see Figure 111)

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

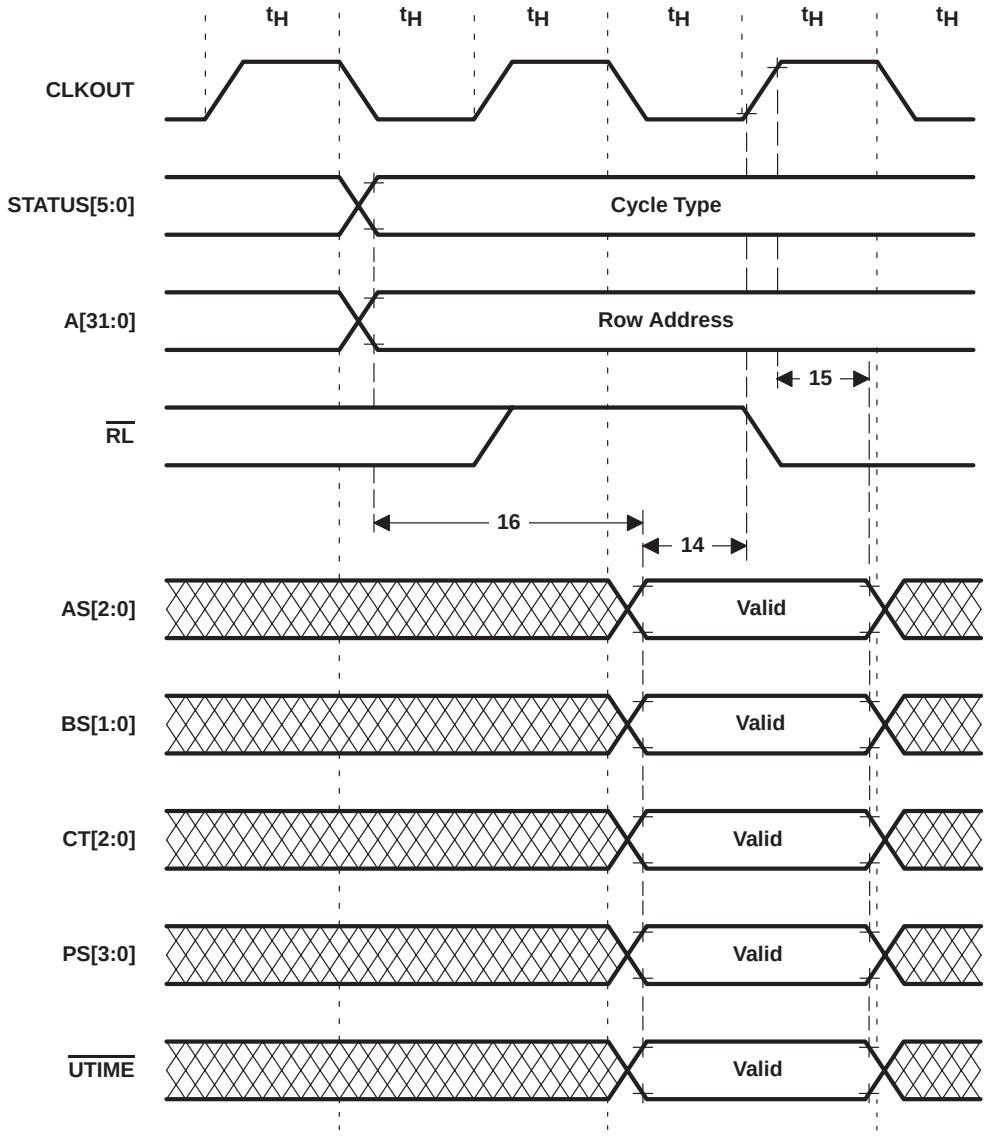


Figure 111. Local Bus Timing: Cycle Configuration Inputs

local bus timing: cycle completion inputs (see Figure 112 and Figure 113)

The cycle completion inputs are sampled at the beginning of each row access at the start of the r3 state. The READY input is sampled also 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				MIN	MAX	UNIT
17	$t_{a(MIDV-CMPV)}$	Access time, $\overline{\text{RETRY}}$, READY, $\overline{\text{FAULT}}$ valid after memory identification (A, STATUS) valid			nt_H-8	ns
18	$t_{su(CMPV-CKOL)}$	Setup time, $\overline{\text{RETRY}}$, READY, $\overline{\text{FAULT}}$ valid to CLKOUT no longer high		7.5		ns
19	$t_{h(CKOL-CMPV)}$	Hold time, $\overline{\text{RETRY}}$, READY, $\overline{\text{FAULT}}$ valid after CLKOUT low		1.2		ns
20	$t_{a(RASL-RRV)}$	Access time $\overline{\text{RETRY}}$, READY valid from $\overline{\text{RAS}}$ low			$nt_H-7.5$	ns
21	$t_{a(RLL-RRV)}$	Access time, $\overline{\text{RETRY}}$, READY valid from $\overline{\text{RL}}$ low			$nt_H-7.5$	ns
22	$t_{a(CASL-RRV)}$	Access time, READY valid from $\overline{\text{CAS}}$ low	2 cyc/col accesses		t_H-12	ns
			3 cyc/col accesses		$2t_H-8$	

local bus timing: cycle completion inputs (continued)

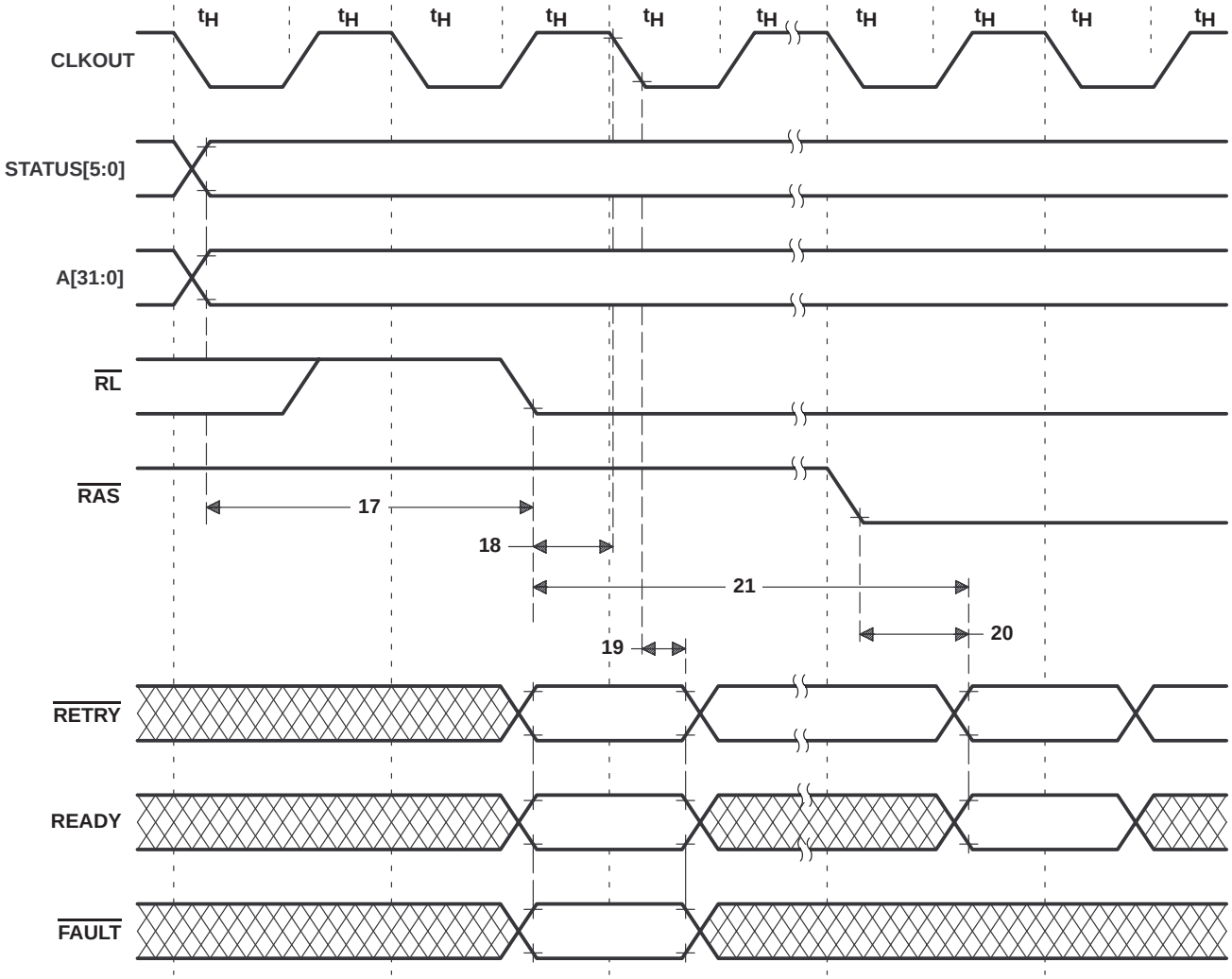


Figure 112. Local Bus Timing: Row-Time Cycle Completion Inputs

local bus timing: cycle completion inputs (continued)

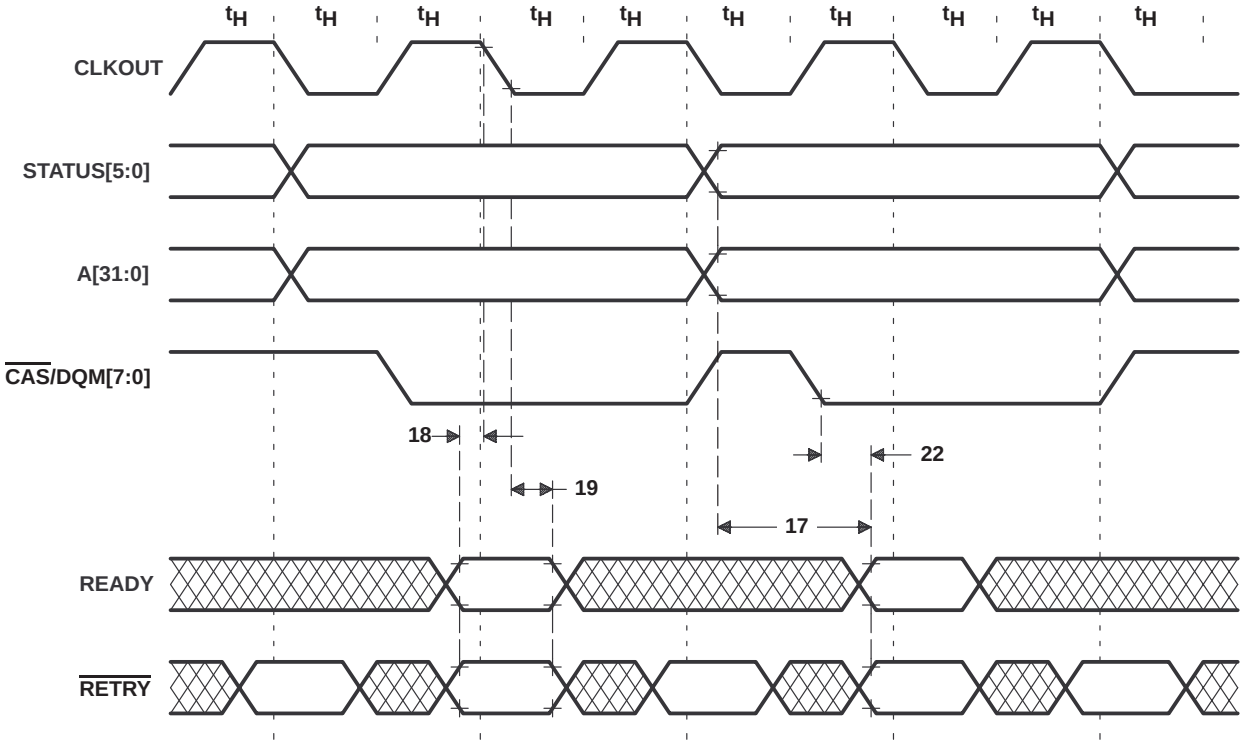


Figure 113. Local Bus Timing: Column-Time Cycle Completion Inputs

general output signal characteristics over operating conditions

The following general timing parameters apply to all SMJ320C80 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 ($\overline{\text{DBEN}}$, $\overline{\text{RL}}$, $\overline{\text{DDIN}}$, DSF, $\overline{\text{RAS}}$, $\overline{\text{TRG}}$/$\overline{\text{CAS}}$, $\overline{\text{W}}$). When measuring output to output, the named group refers to the second output to transition (output B), and the first output (output A) refers to any output group.

NO	PARAMETER		MIN	MAX	UNIT
23	$t_{h(OUTV-CKOL)}$	Hold time, CLKOUT high after output valid D[63:0] A[31:0], STATUS[5:0], $\overline{\text{CAS}}$/DQM[7:0], $\overline{\text{DBEN}}$, $\overline{\text{DDIN}}$, DSF, $\overline{\text{RAS}}$, $\overline{\text{TRG}}$/$\overline{\text{CAS}}$, $\overline{\text{W}}$, $\overline{\text{RL}}$	nt_H–5.6 nt_H–5.0† nt_H–4.3		ns
24	$t_{h(OUTV-CKOH)}$	Hold time, CLKOUT low after output valid D[63:0] A[31:0], STATUS[5:0], $\overline{\text{CAS}}$/DQM[7:0], $\overline{\text{DBEN}}$, $\overline{\text{DDIN}}$, DSF, $\overline{\text{RAS}}$, $\overline{\text{TRG}}$/$\overline{\text{CAS}}$, $\overline{\text{W}}$, $\overline{\text{RL}}$	nt_H–5.6 nt_H–5.0† nt_H–4.3		ns
25	$t_{h(CKOL-OUTV)}$	Hold time, output valid after CLKOUT low	nt_H–5.5		ns
26	$t_{h(CKOH-OUTV)}$	Hold time, output valid after CLKOUT high	nt_H–5		ns
27	$t_{h(OUTV-OUTV)}$	Hold time, output valid after output valid D[63:0] A[31:0], STATUS[5:0], $\overline{\text{CAS}}$/DQM[7:0], $\overline{\text{DBEN}}$, $\overline{\text{DDIN}}$, DSF, $\overline{\text{RAS}}$, $\overline{\text{TRG}}$/$\overline{\text{CAS}}$, $\overline{\text{W}}$, $\overline{\text{RL}}$	nt_H–6.5 nt_H–6.0† nt_H–5		ns
28	$t_{d(CKOH-OUTV)}$	Delay time, CLKOUT no longer low to output valid D[63:0] A[31:0], STATUS[5:0], $\overline{\text{CAS}}$/DQM[7:0], $\overline{\text{DBEN}}$, $\overline{\text{DDIN}}$, DSF, $\overline{\text{RAS}}$,$\overline{\text{TRG}}$/$\overline{\text{CAS}}$, $\overline{\text{W}}$, $\overline{\text{RL}}$		nt_H+6.5 nt_H+5.5† nt_H+5	ns
29	$t_{d(CKOL-OUTV)}$	Delay time, CLKOUT no longer high to output valid D[63:0] A[31:0], STATUS[5:0], $\overline{\text{CAS}}$/DQM[7:0], $\overline{\text{DBEN}}$, $\overline{\text{DDIN}}$, DSF, $\overline{\text{RAS}}$,$\overline{\text{TRG}}$/$\overline{\text{CAS}}$, $\overline{\text{W}}$, $\overline{\text{RL}}$		nt_H+6.5 nt_H+5.5† nt_H+5	ns
30	$t_{d(OUTV-CKOH)}$	Delay time, output no longer valid to CLKOUT high		nt_H+5	ns
31	$t_{d(OUTV-CKOL)}$	Delay time, output no longer valid to CLKOUT low		nt_H+5.5	ns
32	$t_{d(OUTV-OUTV)}$	Delay time, output no longer valid to output valid D[63:0] A[31:0], STATUS[5:0], $\overline{\text{CAS}}$/DQM[7:0], $\overline{\text{DBEN}}$, $\overline{\text{DDIN}}$, DSF, $\overline{\text{RAS}}$, $\overline{\text{TRG}}$/$\overline{\text{CAS}}$, $\overline{\text{W}}$, $\overline{\text{RL}}$		nt_H+6.5 nt_H+6.0† nt_H+5	ns
33	$t_{w(OUTV)}$	Pulse duration, output valid D[63:0] A[31:0], STATUS[5:0], $\overline{\text{CAS}}$/DQM[7:0], $\overline{\text{DBEN}}$, $\overline{\text{DDIN}}$, DSF, $\overline{\text{RAS}}$, $\overline{\text{TRG}}$/$\overline{\text{CAS}}$, $\overline{\text{W}}$, $\overline{\text{RL}}$	nt_H–6.5 nt_H–6.0† nt_H–5.0		ns

† Except for $\overline{\text{CAS}}$/DQM[7:0] during nonuser-timed 2-cycle/column accesses

general output signal characteristics over operating conditions (continued)

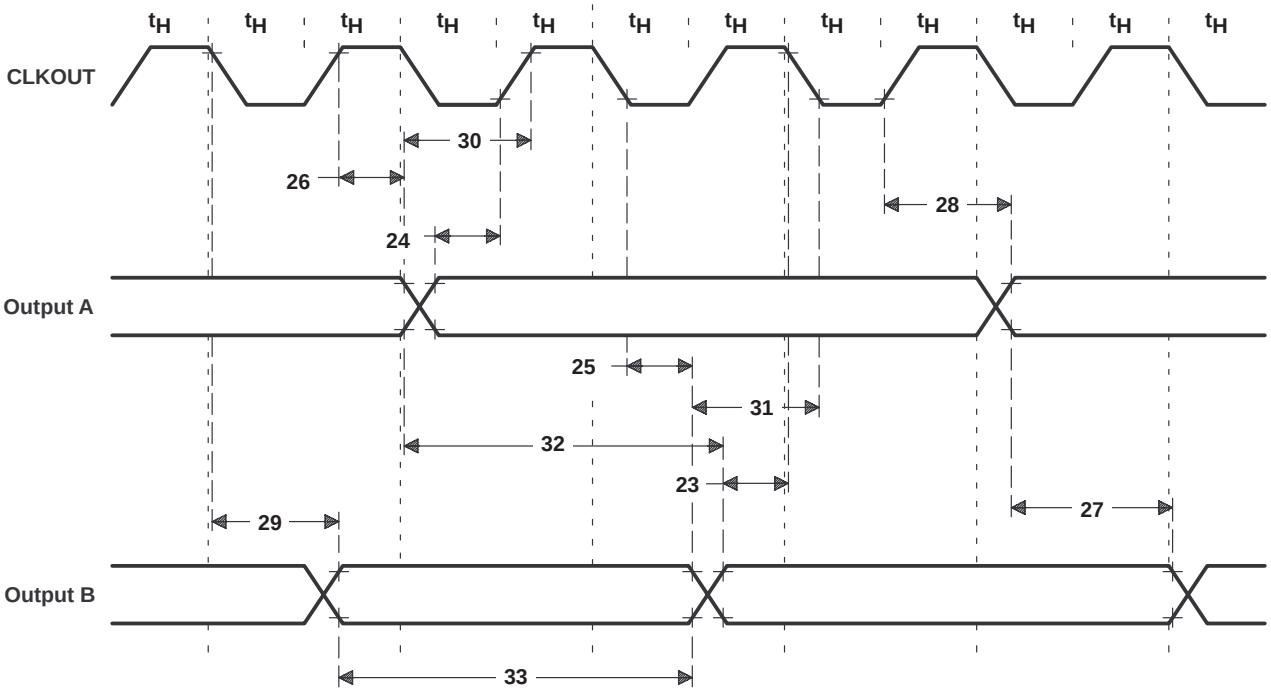


Figure 114. General Output-Signal Timing

data input timing (see Figure 115)

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.

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

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

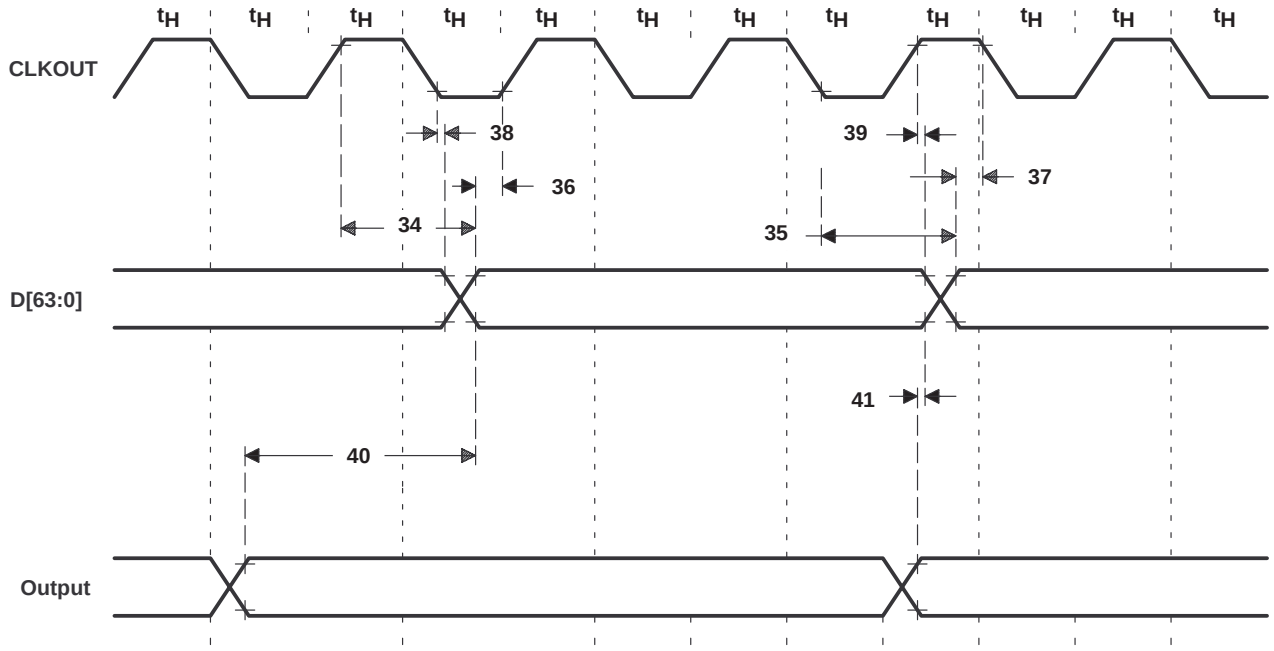


Figure 115. Data-Input Timing

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

These timing parameters apply to the $\overline{\text{CAS}}$/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 (nonuser-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.

NO			MIN	MAX	UNIT
42	$t_{w(CASH)}$	Pulse duration, $\overline{\text{CAS}}$/DQM high	t_H–2		ns
43	$t_{w(CASL)}$	Pulse duration, $\overline{\text{CAS}}$/DQM low	$3t_H$–9.5		ns
44	$t_{h(OUTV-CASL)}$	Hold time, $\overline{\text{CAS}}$/DQM high after output valid D[63:0] A[31:0], STATUS[5:0] $\overline{\text{DBEN}}$, $\overline{\text{DDIN}}$, DSF, $\overline{\text{RAS}}$, $\overline{\text{RL}}$, $\overline{\text{TRG}}$/$\overline{\text{CAS}}$, $\overline{\text{W}}$	nt_H–4.5 nt_H–4.0 nt_H–3		ns
45	$t_{h(CASL-OUTV)}$	Hold time, output valid after $\overline{\text{CAS}}$/DQM low	nt_H–9.5		ns
46	$t_{a(CASL-DV)}$	Access time, data valid from $\overline{\text{CAS}}$/DQM low		$3t_H$–12	ns
47	$t_{h(CASH-DV)}$	Hold time, data valid after $\overline{\text{CAS}}$/DQM high	2		ns

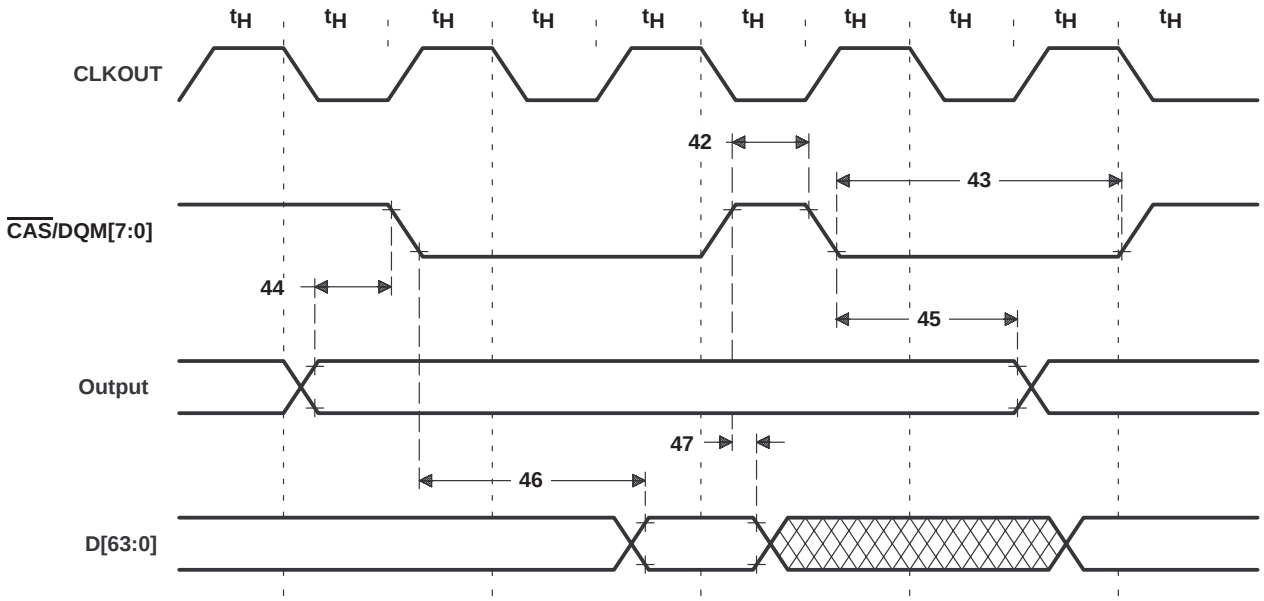


Figure 116. 2-Cycle/Column $\overline{\text{CAS}}$ Timing

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

external interrupt timing (see Figure 117)

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

NO			MIN	MAX	UNIT
48	$t_{w(EINL)}$	Pulse duration, $\overline{\text{EINTx}}$ low	6*		ns
49	$t_{su(EINH-CKOH)}$	Setup time, $\overline{\text{EINTx}}$ high before CLKOUT no longer low	9.5†		ns
50	$t_{w(EINH)}$	Pulse duration, $\overline{\text{EINTx}}$ high	6*		ns
51	$t_{su(LINL-CKOL)}$	Setup time, $\overline{\text{LINT4}}$ low before CLKOUT no longer high	9.5†		ns

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

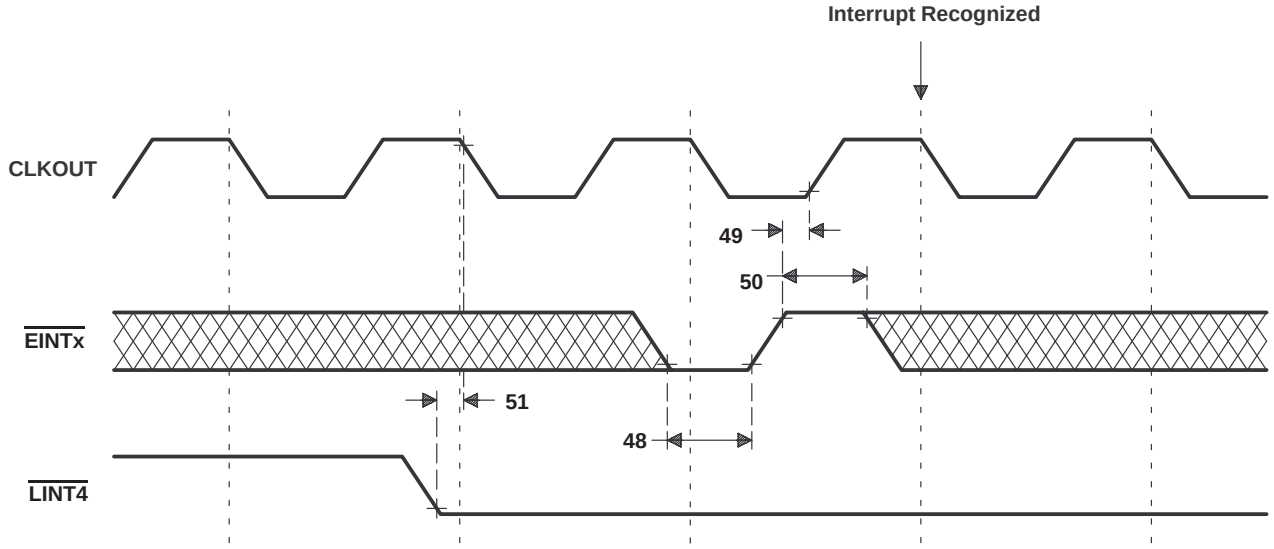


Figure 117. External Interrupt Timing

XPT input timing (see Figure 118 and Figure 119)

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

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

† 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 not production tested.

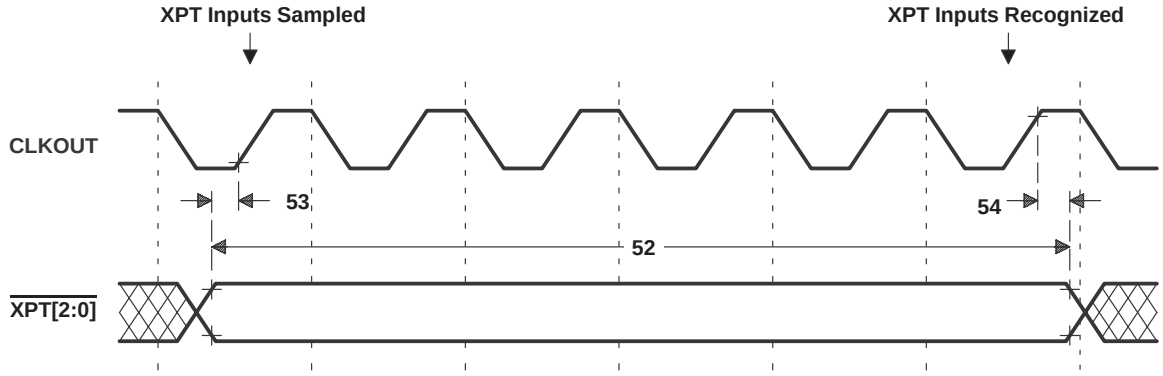


Figure 118. XPT Input Timing – XPT Recognition

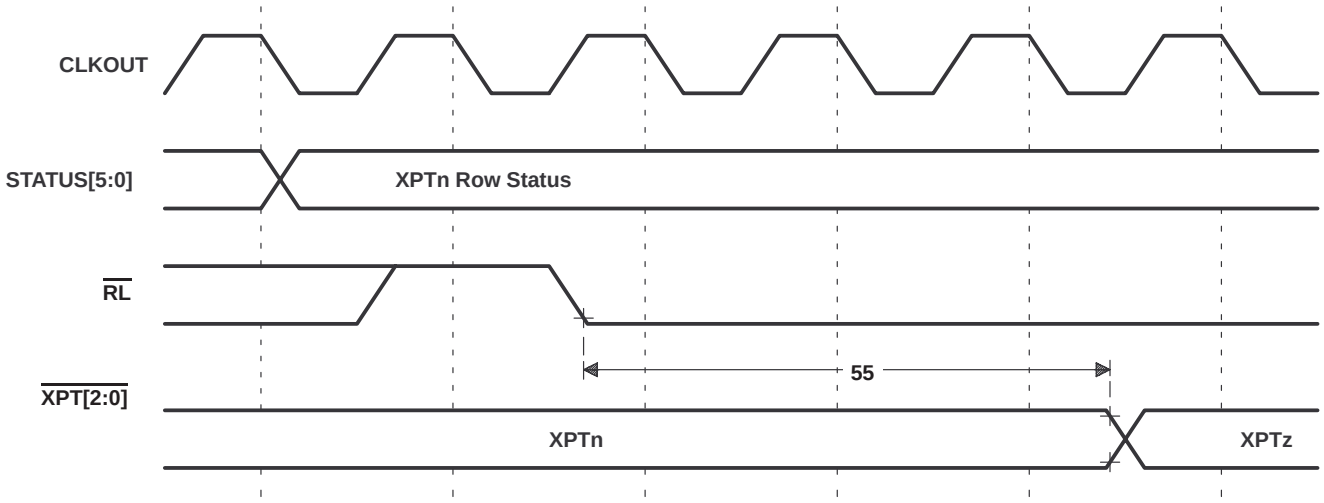


Figure 119. XPT Input Timing – XPT Service

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

host-interface timing (see Figure 120)

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

* This parameter is not production tested.

NOTE 5: Parameter must be met only to ensure $\overline{HREQ}$ recognition during the indicated clock cycle.

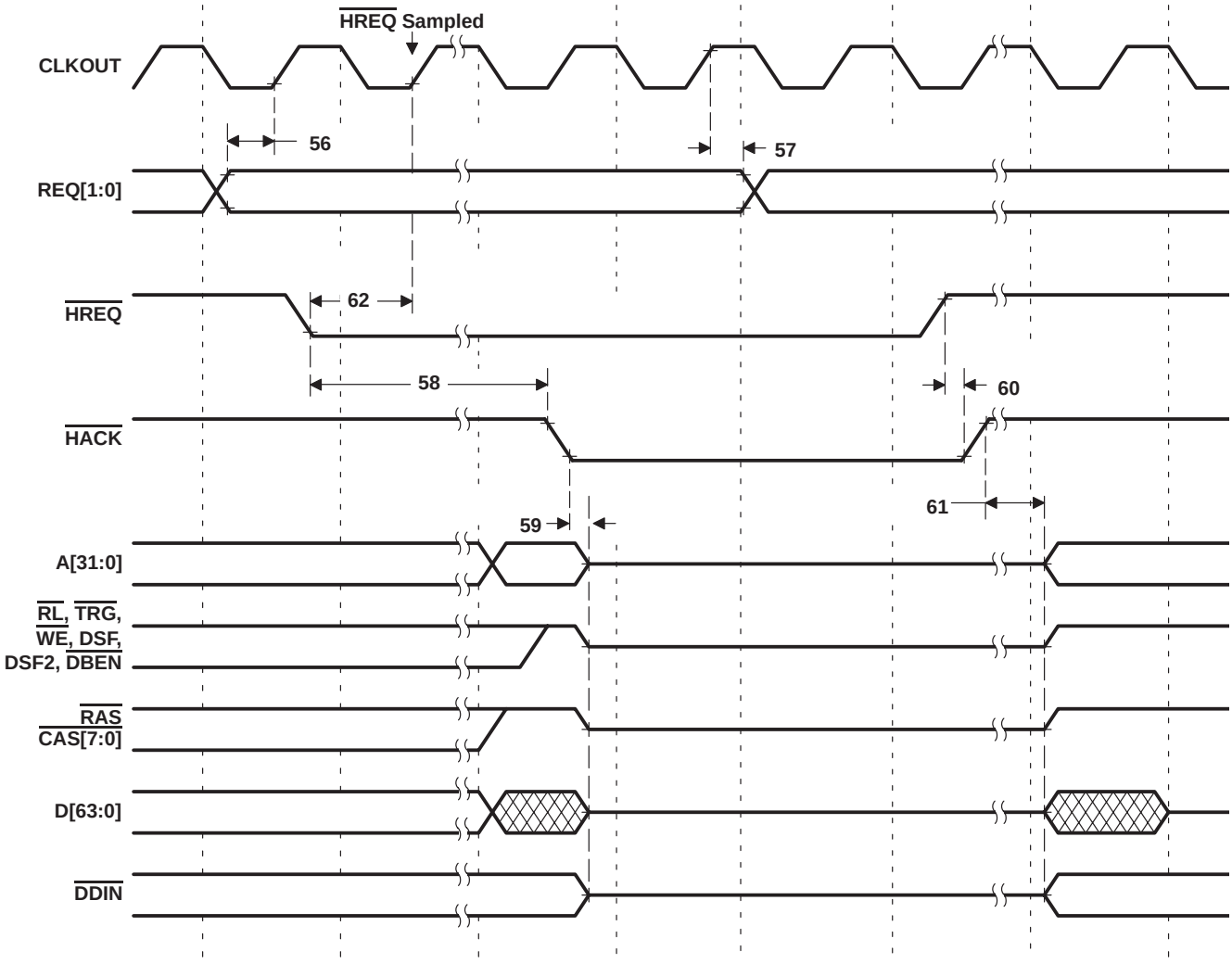


Figure 120. Host-Interface Timing

video interface timing: SCLK timing (see Figure 121)

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 not production tested.

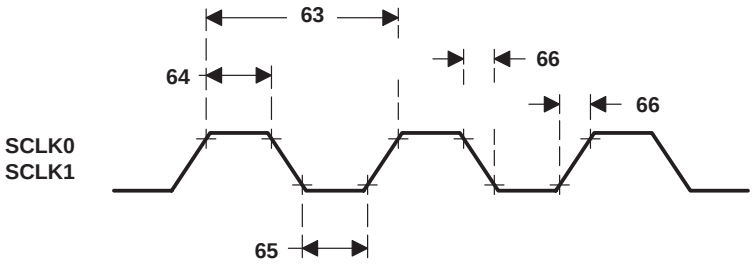


Figure 121. Video Interface Timing: SCLK Timing

video interface timing: FCLK input and video outputs (see Note 6 and Figure 122)

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, $\overline{\text{HSYNC}}$, $\overline{\text{VSYNC}}$, $\overline{\text{CSYNC}}$/$\overline{\text{HBLNK}}$, $\overline{\text{CBLNK}}$/$\overline{\text{VBLNK}}$, or CAREA high after FCLK low	0		ns
72	$t_{h(FCKL-SYH)}$	Hold time, $\overline{\text{HSYNC}}$, $\overline{\text{VSYNC}}$, $\overline{\text{CSYNC}}$/$\overline{\text{HBLNK}}$, $\overline{\text{CBLNK}}$/$\overline{\text{VBLNK}}$, or CAREA low after FCLK low	0		ns
73	$t_{d(FCKL-SYL)}$	Delay time, FCLK no longer high to $\overline{\text{HSYNC}}$, $\overline{\text{VSYNC}}$, $\overline{\text{CSYNC}}$/$\overline{\text{HBLNK}}$, $\overline{\text{CBLNK}}$/$\overline{\text{VBLNK}}$, or CAREA low		20	ns
74	$t_{d(FCKL-SYH)}$	Delay time, FCLK no longer high to $\overline{\text{HSYNC}}$, $\overline{\text{VSYNC}}$, $\overline{\text{CSYNC}}$/$\overline{\text{HBLNK}}$, $\overline{\text{CBLNK}}$/$\overline{\text{VBLNK}}$, or CAREA high		20	ns

* This parameter is not production 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.

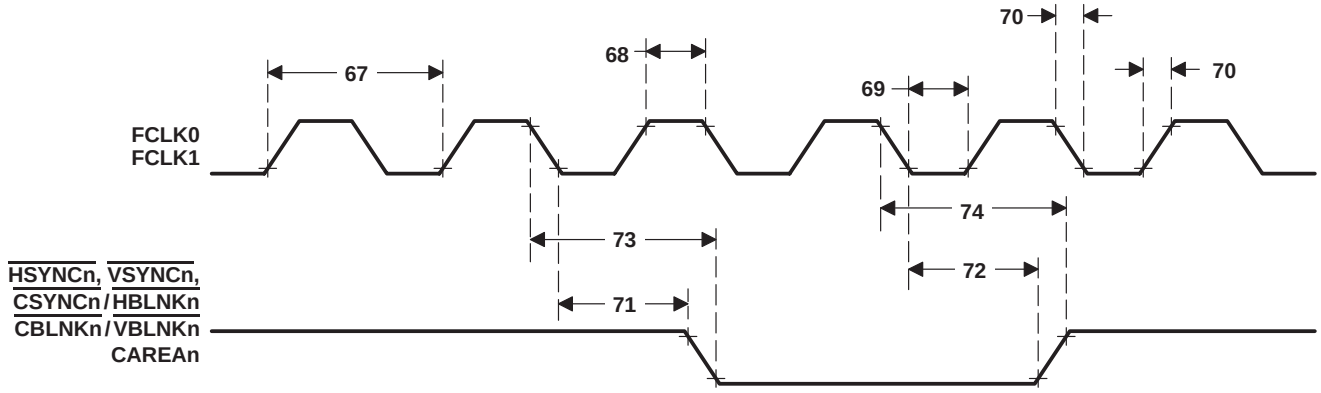


Figure 122. Video Interface Timing: FCLK Input and Video Outputs

video interface timing: external sync inputs (see Figure 123)

When configured as inputs, the $\overline{\text{HSYNCn}}$, $\overline{\text{VSYNCn}}$, and $\overline{\text{CSYNCn}}$ 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.

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

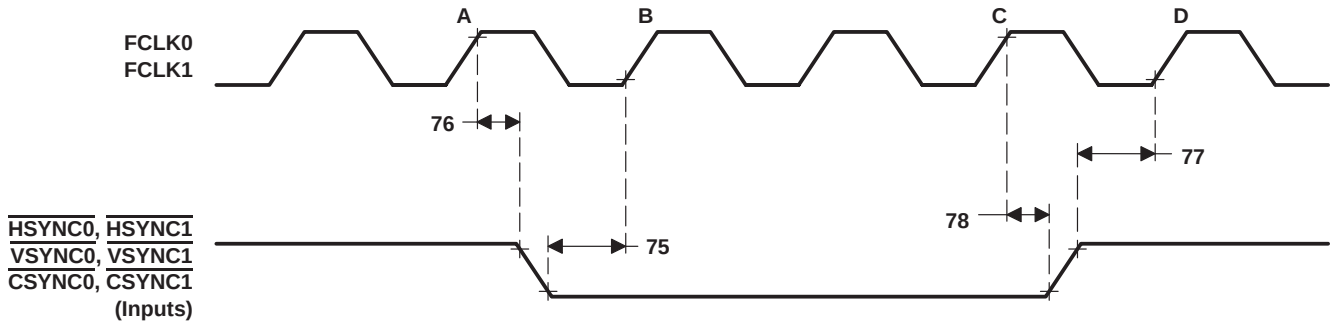


Figure 123. Video Interface Timing: External Sync Inputs

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

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 124. Table 38 describes the emulation signal.

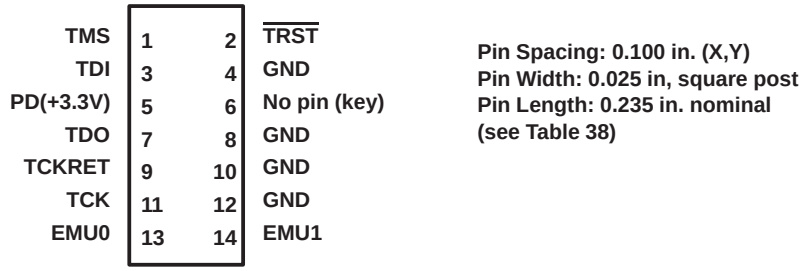


Figure 124. Target System Header

Table 38. Target Connectors

XDS510™ SIGNAL	XDS510 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.†
$\overline{\text{TRST}}$	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 XDS510 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 125.

XDS510 is a trademark of Texas Instruments Incorporated.

emulator-interface connection (continued)

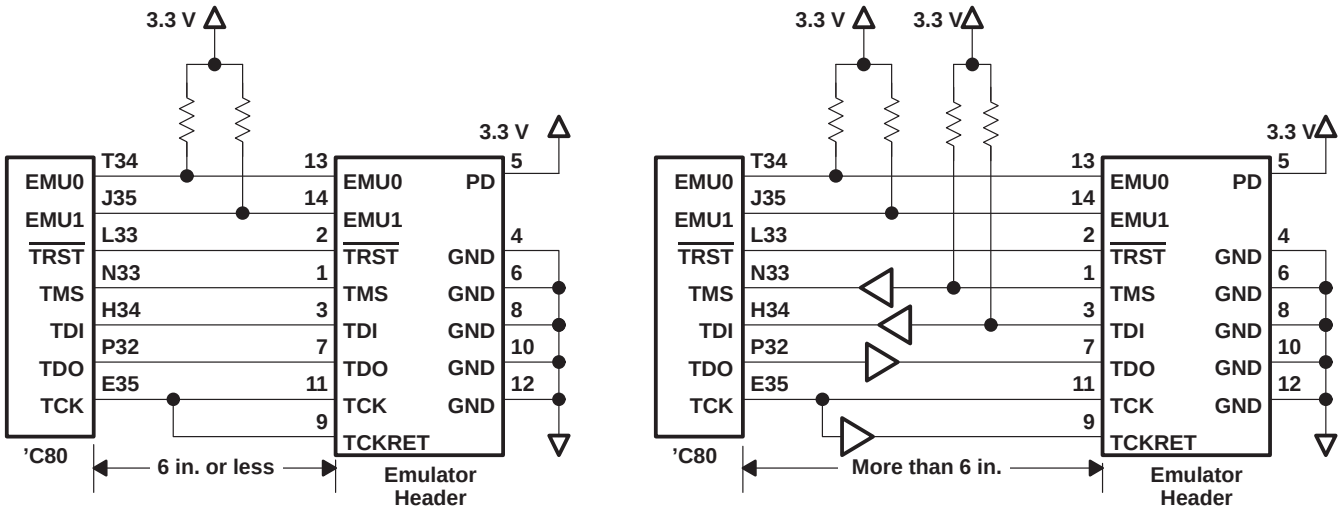


Figure 125. Emulation Header Connections – Emulator-Driven Test Clock

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

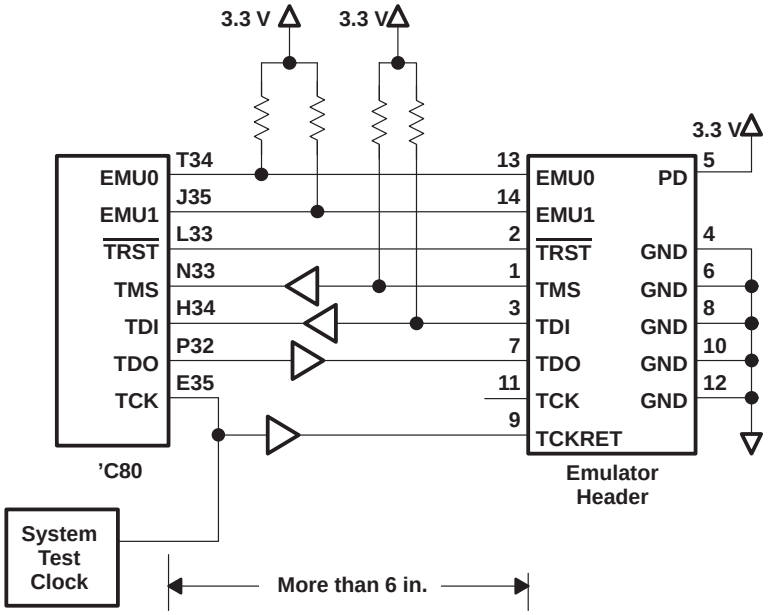


Figure 126. Emulation Header Connections – System-Driven Test Clock

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 recommended highly 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.

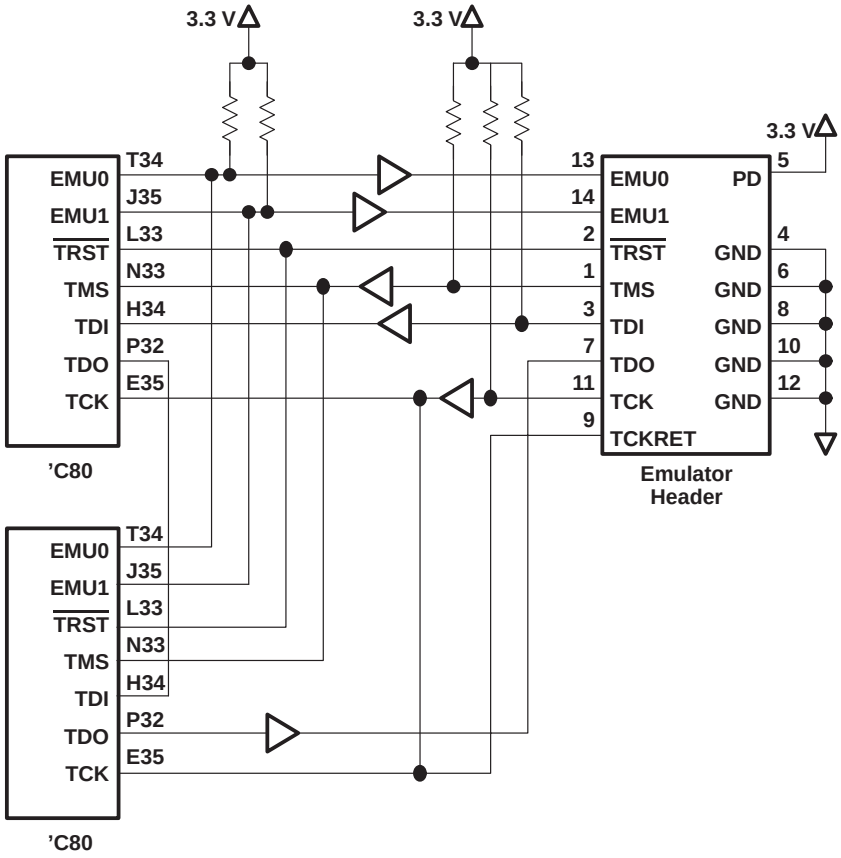
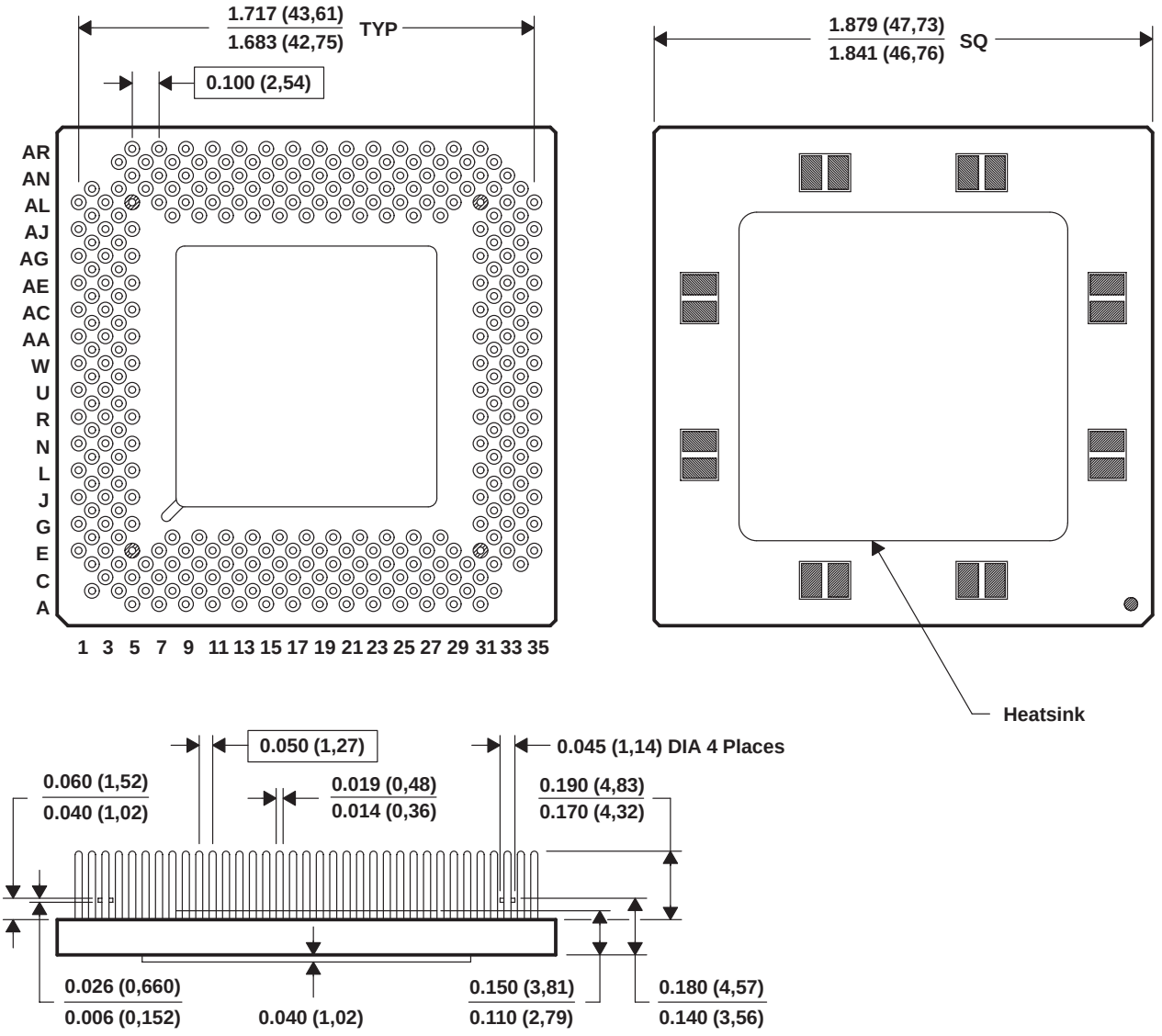


Figure 127. Emulation Header Connections – Multiprocessor Applications

MECHANICAL DATA

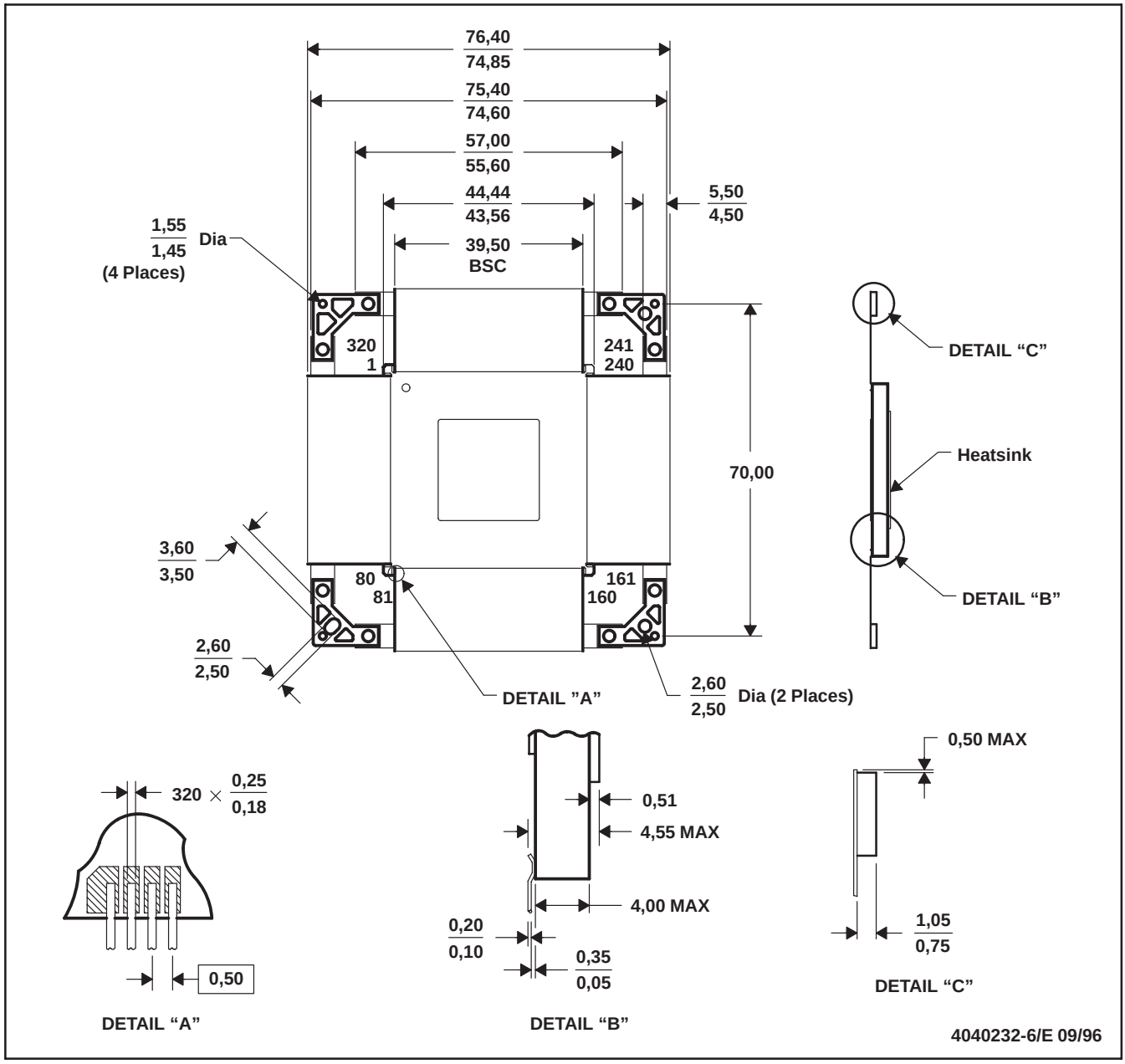
GF (S-CPGA-P305) **CERAMIC PIN GRID ARRAY PACKAGE**



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

HFH (R-CQFP-F320) **CERAMIC QUAD FLATPACK WITH NCTB**



NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. This package can be hermetically sealed with a metal lid.
D. The terminals will be gold plated.
E. Falls within JEDEC MO-134 AD

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products 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, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

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

TI assumes no liability for applications assistance or customer product design. TI does not 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. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.