LH77790A
Embedded Microcontroller

User's Guide

Version 3.0

SHARP

SHARP reserves the right to make changes in specifications described herein at any time and without notice in order to improve design or reliability. SHARP does not assume any responsibility for the use of any circuitry described; no circuit patent licenses are implied. SHARP assumes no responsibility for damage caused by misuse or improper use of devices.

LIFE SUPPORT POLICY

SHARP components should not be used in medical devices with life support functions, safety equipment (or similar applications where component failure would result in loss of life or physical harm), aerospace equipment, telecommunication equipment (trunk lines) or nuclear power control equipment. Contact a SHARP representative or sales office before using SHARP devices for any applications other than those recommended by SHARP.

WARRANTY

SHARP warrants to Customer that the Products will be free from defects in material and workmanship under normal use and service for a period of one year from the date of invoice. Customer's exclusive remedy for breach of this warranty is that SHARP will either (i) repair or replace, at its option, any Product which fails during the warranty period because of such defect (if Customer promptly reported the failure to SHARP in writing) or, (ii) if SHARP is unable to repair or replace, SHARP will refund the purchase price of the Product upon its return to SHARP. This warranty does not apply to any Product which has been subjected to misuse, abnormal service or handling, or which has been altered or modified in design or construction, or which has been serviced or repaired by anyone other than SHARP. The warranties set forth herein are in lieu of, and exclusive of, all other warranties, express or implies. ALL EXPRESS AND IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR USE AND FITNESS FOR A PARTICULAR PURPOSE ARE SPECIFICALLY EXCLUDED.

LH77790A Embedded Microcontroller User's Guide Version 3.0
 Ref. No. SMT96100-B

TABLE OF CONTENTS

Chapter 1

OVERVIEW

INTRODUCTION

Portable devices are becoming more and more prevalent in our daily life. They are used as personal information managers, communication devices, digital cameras, handheld games, bar-code scanners, medical equipment, electronic instrumentation, and navigation systems. There are significant design challenges for portable devices. Low cost is a top priority for high volume products. Low power is a must for long battery life. High performance is critical for computationally-intensive applications such as PDAs, GPS, and 2-D scanners. Communication capabilities and effective user interface are integral parts of any portable device. Last, but not least, superior product development support tools are crucial to reducing time-to-market.

The SYSTEM ON CHIP Business Unit at Sharp has designed the LH77790A Embedded Microcontroller (a.k.a. 790A) to meet the above challenges in portable design. The LH77790A, powered by an ARM7DI™, is a complete system on chip with a high level of integration to satisfy a wide range of customer requirements and expectations.

The 790A combines a 32-bit ARM7DI RISC engine, a number of essential peripherals (UARTs, Counter/Timers, PIOs, PWMs, etc...), LCD controller, cache, and on-chip SRAM. This high level of integration lowers overall system cost, reduces development cycle time and accelerates product introduction. The 790A's fully static design, power management unit, dual voltage operation (3.3 V/5 V), fast interrupt response time, on-chip cache and SRAM, powerful instruction set, and very low power RISC core provide high performance at low current draw. The on-chip LCD controller, UARTs, IrDA/DASK, and the programmable peripheral interface (PPI) are well suited for wireless, cable, and visual communication requirements. Other features like, watchdog timer, programmable memory interface, on-chip SRAM/DRAM controllers and debug support provides a high level of flexibility.

Please check our website at www.sharpmeg.com or with your local SHARP sales office for the latest Thermal and Electrical Specifications and/or errata sheets. These documents will contain the latest parameters for the LH77790A.

FEATURES

- Highly Integrated Single Chip
- 32-Bit ARM7DI RISC Core
  - Built-In Debug and ICE Support
  - Fast Interrupt Response
  - Powerful Instruction Set
- 26-bit External Address Bus
  - 512MB Addressable Space
- 16-bit External Data Bus
- 2KB Data/Instruction Cache
  - 4 Way Set Associative
  - Write Back Policy
  - Flexible Modes of Operation
- 2KB Static RAM
  - Expandable to 4KB Without Cache
- Low Power
- High Performance
- Programmable Clock and Power Management
- Programmable Monochrome LCD Controller
  - 1024 (V) × 2048 (H)
  - Four Gray Shades
  - Frame buffer in Main Memory
- On-Chip Interrupt Controller
  - Six External Interrupts
  - Seven Internal Interrupts
  - ARM7DI Wake-Up
- Three UARTs - 16C450-class
  - Full Modem Support on UART0
  - Partial Modem Support on UART1
  - IrDA-1.0/DASK Support on UART2
- IrDA/DASK IR Interface
  - IrDA-1.0 (2.4 kbps to 115.2 kbps)
  - DASK (2.4 kbps to 57.6 kbps)
- Three Pulse Width Modulator Channels
  - PWM0 and PWM1 have 8-Bit Resolution
  - PWM2 has 16-Bit Resolution
- Flexible Memory Interface
  - Six Multiplexeled Chip Enables/ CAS pins
  - Two RAS pins
  - Fully Programmable
  - Six SRAM Banks (64MB each)
  - Two DRAM Banks (128MB each)
  - Access Privileges (System/User)
- On-Chip DRAM Controller
  - Fast Page Mode
  - Normal Mode
  - CAS before RAS Refresh
- Programmable Peripheral Interface (PPI)
  - 24 Programmable I/O Signals
  - Three Modes of Operation
- Three 16-Bit Counter/Timer Channels
  - Six Modes of Operation
  - Binary or BCD Counting
- Hardware Watchdog Timer
  - Eight Time-out Intervals
  - Protection Mechanism
  - Three Time-out Actions
- Little Endian
- JTAG Interface
- Dual Supply Voltage
  - 5 V TTL - 25 MHz
  - 3.3 V LVTTL - 16.7 MHz

DEVELOPMENT ENVIRONMENT

The 790A Evaluation Board (part number LU7790AH2A) and the ARM Software Development ToolKit (part number LU7V211H1) give users full access to the power and features of the 790A and provide a complete integrated environment for development. Users will be able to develop, benchmark, and profile both hardware and software easily and quickly.

BLOCK DIAGRAM

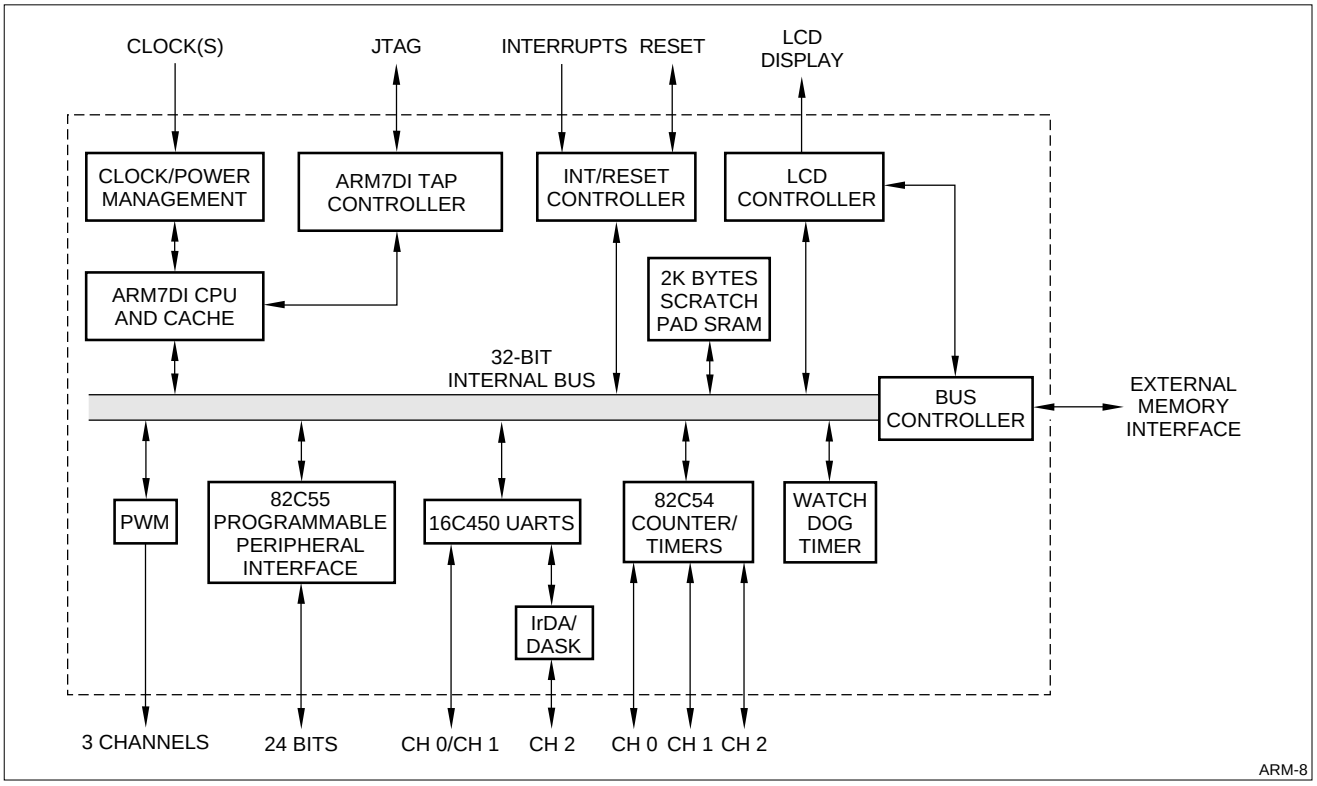


Figure 1-1. LH77790A Block Diagram

CONVENTION

Table 1-1. Title Needed

CONVENTION	DESCRIPTION	EXAMPLE
0x	Hex Number Prefix	0xFFFFA400 0x30
0b	Binary Number Prefix*	0b01101 01101*
X	Don't Care Value	0b0110X

NOTE: * Sometimes a binary number is written with-out '0b' prefix.

ACCESSING REGISTERS

All registers that are more than 8-bit wide should be treated as 32-bit registers. Otherwise, the compiler will access them using two STRB/LDRB instructions (Store/Load Byte) resulting in incorrect data being written or read.

Chapter 2
PIN DESCRIPTION

Table 2-1.
Pin Description

PIN(s)	NAME	DIRECTION	DESCRIPTION
EXTERNAL BUS INTERFACE			
36 - 31, 28 - 21, 18 - 11, 8 - 5	A[25:0]	O	External Address bus. The 790A will provide a 26-bit address to external memories and peripherals.
60 - 55, 52 - 47, 42 - 41, 38 - 37	D[15:0]	I/O	External 16-Bit data bus.
72	$\overline{OE}$	O	Output Enable for external memory and peripherals. $\overline{OE}$ allows external memory and peripherals to drive the data bus and is asserted LOW during a read access and HIGH during a write access.
71	$\overline{WE}$	O	Write Enable for external memory and peripherals. During a write access, this pin is driven LOW. During a read access, this pin is driven HIGH.
70 - 65	$\overline{CE}$ [5:0]/ $\overline{CAS}$ [5:0]	O	These pins provide the Chip Enable/Column Address Select signals allowing direct connection to standard external memory/peripheral devices. The pins act as $\overline{CAS}$ when interfacing to DRAMs and as $\overline{CE}$ otherwise. They are fully programmable by the system designer and can support byte enables.
62 - 61	$\overline{RAS}$ [1:0]	O	Row Address Select pins for DRAM Bank 0 and Bank 1.
74	$\overline{WAIT}$	I	External Memory Wait. Allows the use of slow memories. The 790A generates external $\overline{WAIT}$ cycles (EWC) in response to activating $\overline{WAIT}$. $\overline{WAIT}$ is sampled on the HIGH to LOW transition on XCLK. To add one EWC, $\overline{WAIT}$ must be active prior to sampling in the last cycle (beginning of the last cycle) of a memory transfer. If $\overline{WAIT}$ continues to be active (when sampled) in subsequent cycles, more EWC will be added. Once $\overline{WAIT}$ is deactivated, the 790A will complete the memory transfer.
73	$\overline{BW}$	O	Byte Wide Access. $\overline{BW}$ is LOW when the ARM7DI executes a store/load byte instruction. $\overline{BW}$ is HIGH when the ARM7DI Core executes a store/load word instruction or an instruction fetch. $\overline{BW}$ does not depend on the bus size of the external memory/peripheral device. $\overline{BW}$ is valid during an external memory access. It can be used by an external address decoder to generate extra chip/byte enables. $\overline{BW}$ is a don't care during DRAM refresh.
169	$\overline{BB}$	I	Byte Boot selects between x8 or x16 for the boot memory. The 790A samples and captures the state of $\overline{BB}$ on the rising edge of $\overline{RESETI}$ allowing $\overline{BB}$ to change state after Reset. If $\overline{BB}$ is LOW the 790A will boot from a x8 memory. If $\overline{BB}$ is HIGH, the 790A will boot from a x16 memory. This pin is normally tied LOW for x8 boot memory or HIGH for x16 boot memory.
COUNTERS/TIMERS INTERFACE			
123, 121, 117	CTGATE[2:0]	I	Counter/Timer control gate input signals.
124, 122, 118	CTOUT[2:0]	O	Counter/Timer output signals.
INTERRUPT INTERFACE			
107 - 102	INT[5:0]	I	External interrupt input signals.

Table 2-1.
Pin Description (cont'd)

PIN(s)	NAME	DIRECTION	DESCRIPTION
LCD CONTROLLER INTERFACE			
91	CP2	O	Shift/Pixel Clock.
92	CP1	O	Line Pulse/HSYNC.
93	MCLK	O	AC Modulation Signal.
94	S	O	Frame Pulse/VSYNC.
95	LCDCNTL	O	LCD Control Signal.
84 - 77	VD[7:0]	O	Video Data.
PROGRAMMABLE PERIPHERAL INTERFACE			
139 - 135, 128 - 126 149 - 145, 142 - 140 159 - 155, 152 - 150	PA[7:0] PB[7:0] PC[7:0]	I/O	Parallel ports A, B, and C signals. Signals have programmable access and can function as Input, Output or Controls (port C only). PB[7:2] and PC[2:0] are multiplexed with UART's modem signals
PWM INTERFACE			
98 - 96	PWM[2:0]	O	Pulse Width Modulator output signals.
UARTs INTERFACE			
114, 112, 108	RxD[2:0]	I	UART serial data input signals. RxD2 also doubles as the digital input for the IR interface.
115, 113, 111	TxD[2:0]	O	UART serial data output signals. TxD2 also doubles as the digital output for the IR interface.
150, 151	$\overline{\text{RTS}}$ [1:0]	O	Request To Send for UART0 and UART1. Multiplexed with PC0 and PC1 respectively.
145, 146	$\overline{\text{CTS}}$ [1:0]	I	Clear To Send for UART0 and UART1. Multiplexed with PB3 and PB4 respectively.
142, 147	$\overline{\text{RI}}$ [1:0]	I	Ring Indicator for UART0 and UART1. Multiplexed with PB2 and PB5 respectively.
152	$\overline{\text{DTR0}}$	O	Data Terminal Ready for UART0 only. Multiplexed with PC2.
149	$\overline{\text{DSR0}}$	I	Data Set Ready for UART0 only. Multiplexed with PB7.
148	$\overline{\text{DCD0}}$	I	Data Carrier Detect for UART0 only. Multiplexed with PB6.
RESET AND EXTERNAL CLOCKS			
101	$\overline{\text{RESETI}}$ **	I	Chip and JTAG TAP Controller Reset Input. $\overline{\text{RESETI}}$ has a built-in glitch detector. $\overline{\text{RESET0}}$ will be driven LOW after a valid reset is detected for as long as $\overline{\text{RESETI}}$ is driven LOW. JTAG reset, $\overline{\text{TRST}}$, is internally connected to $\overline{\text{RESETI}}$.
119	$\overline{\text{RESET0}}$	O	Chip Reset Output. It will be driven LOW during: (1) Chip Reset (2) WDT Timeout Reset (3) Software Controlled Reset
3	XCLK	I	The 790A External Clock Input pin. Duty cycle is 50%.

NOTE: **JTAG Reset, $\overline{\text{TRST}}$, is internally connected to $\overline{\text{RESETI}}$. IEEE 1149.1 – 1990 Standard requires JTAG Inputs to be pulled up to a good logic level to achieve normal operations.

Table 2-1.
Pin Description (cont'd)

PIN(s)	NAME	DIRECTION	DESCRIPTION
RESET AND EXTERNAL CLOCKS (cont'd)			
162	XCLKDIS	O	XCLKDIS is an active HIGH output pin that can be used to disable external clock circuitry and will result in reducing current consumption to micro-amperes. XCLKDIS is HIGH in Sleep and Stop modes. Connecting this pin to the external clock circuitry, allows the 790A to go into Stop mode by disabling the external clock.
116	UCLK	I	UART/DASK Demodulator External clock input signal. Duty cycle is 50%.
125	CTCLK	I	Counter/Timer External clock input signal. Duty cycle is 50%.
JTAG INTERFACE**			
160	TCK	I	JTAG Test/EmbeddedICE™ clock input signal. Must be pulled-up for normal operation (56 kΩ is recommended for compatibility with ARM's EmbeddedICE)
161	TMS	I	JTAG Test/EmbeddedICE mode select input signal. Must be pulled-up for normal operation (56 kΩ is recommended for compatibility with ARM's EmbeddedICE)
165	TDI	I	JTAG Test/EmbeddedICE data input signal. Must be pulled-up for normal operation (56 kΩ is recommended for compatibility with ARM's EmbeddedICE)
166	TDO	O	JTAG Test/EmbeddedICE data output signal. Must be pulled-up for normal operation (56 kΩ is recommended for compatibility with ARM's EmbeddedICE)
RESERVED INTERFACE			
170	ADBE	I	Reserved. Must be tied HIGH for normal operation.
167	TEST0	I	Reserved. Must be tied LOW for normal operation.
168	TEST1	O	Reserved. No Connect.
171	TEST2	I	Reserved. Must be tied LOW for normal operation
172	TEST3	O	Reserved. No Connect
POWER SIGNALS			
9, 19, 29, 39, 53, 63, 75, 85, 99, 109, 129, 143, 153, 163, 173	V_{CC}	I	Power. All LH77790A are 5 V/3.3 V.
4, 10, 20, 30, 40, 54, 64, 76, 86, 100, 110, 120, 130, 144, 154, 164, 174	V_{SS}	I	Ground. All ground pins must be used.
NO CONNECT			
1, 2, 43, 44, 45, 46, 87, 88, 89, 90, 131, 132, 133, 134, 175, 176	NC	—	No Connects.

NOTE: **JTAG Reset, $\overline{\text{TRST}}$, is internally connected to $\overline{\text{RESETI}}$.

Table 2-2.
Pinout

PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL
1	NC	45	NC	89	NC	133	NC
2	NC	46	NC	90	NC	134	NC
3	XCLK	47	D4	91	CP2	135	PA3
4	V_{SS}	48	D5	92	CP1	136	PA4
5	A0	49	D6	93	MCLK	137	PA5
6	A1	50	D7	94	S	138	PA6
7	A2	51	D8	95	LCDCNTL	139	PA7
8	A3	52	D9	96	PWM0	140	PB0
9	V_{CC}	53	VCC	97	PWM1	141	PB1
10	V_{SS}	54	VSS	98	PWM2	142	PB2/$\overline{RII}$
11	A4	55	D10	99	V_{CC}	143	V_{CC}
12	A5	56	D11	100	V_{SS}	144	V_{SS}
13	A6	57	D12	101	$\overline{RESETI}$	145	PB3/$\overline{CTS1}$
14	A7	58	D13	102	INT0	146	PB4/$\overline{CTS0}$
15	A8	59	D14	103	INT1	147	PB5/$\overline{RI0}$
16	A9	60	D15	104	INT2	148	PB6/$\overline{DCD0}$
17	A10	61	$\overline{RAS0}$	105	INT3	149	PB7/$\overline{DSR0}$
18	A11	62	$\overline{RAS1}$	106	INT4	150	PC0/$\overline{RTS1}$
19	V_{CC}	63	V_{CC}	107	INT5	151	PC1/$\overline{RTS0}$
20	V_{SS}	64	V_{SS}	108	RxD0	152	PC2/$\overline{DTR0}$
21	A12	65	$\overline{CE0}$/$\overline{CAS0}$	109	V_{CC}	153	V_{CC}
22	A13	66	$\overline{CE1}$/$\overline{CAS1}$	110	V_{SS}	154	V_{SS}
23	A14	67	$\overline{CE2}$/$\overline{CAS2}$	111	TxD0	155	PC3
24	A15	68	$\overline{CE3}$/$\overline{CAS3}$	112	RxD1	156	PC4
25	A16	69	$\overline{CE4}$/$\overline{CAS4}$	113	TxD1	157	PC5
26	A17	70	$\overline{CE5}$/$\overline{CAS5}$	114	RxD2	158	PC6
27	A18	71	$\overline{WE}$	115	TxD2	159	PC7
28	A19	72	$\overline{OE}$	116	UCLK	160	TCK
29	V_{CC}	73	$\overline{BW}$	117	CTGATE0	161	TMS
30	V_{SS}	74	$\overline{WAIT}$	118	CTOUT0	162	XCLKDIS
31	A20	75	V_{CC}	119	$\overline{RESETO}$	163	V_{CC}
32	A21	76	V_{SS}	120	V_{SS}	164	V_{SS}
33	A22	77	VD0	121	CTGATE1	165	TDI
34	A23	78	VD1	122	CTOUT1	166	TDO
35	A24	79	VD2	123	CTGATE2	167	TEST0
36	A25	80	VD3	124	CTOUT2	168	TEST1
37	D0	81	VD4	125	CTCLK	169	$\overline{BB}$
38	D1	82	VD5	126	PA0	170	ADBE
39	V_{CC}	83	VD6	127	PA1	171	TEST2
40	V_{SS}	84	VD7	128	PA2	172	TEST3
41	D2	85	V_{CC}	129	V_{CC}	173	V_{CC}
42	D3	86	V_{SS}	130	V_{SS}	174	V_{SS}
43	NC	87	NC	131	NC	175	NC
44	NC	88	NC	132	NC	176	NC

Chapter 3

ARM7DI CORE

INTRODUCTION

The LH77790A is built around the ARM7DI core. The ARM7DI is comprised of the ARM7 processor, with Debugging (D), In-Circuit-Emulation (I) and JTAG-style port to ease the development of application software, operating systems, and hardware.

FEATURES

- 32-Bit RISC ARM7 CPU
- Built-In Debug (D) and In-Circuit-Emulation (I) Support
- Low Power Consumption
- High Performance
- Fast Interrupt Response with Minimal Context Switching
- 25 MHz at 5 V, 16.7 MHz at 3.3 V
- Powerful Instruction Set
- Little Endian Mode
- 32-Bit Operation Mode
- Fully Static Design for Power Sensitive Applications
- JTAG-style port (not 100% compatible with the IEEE Standard 1149.1 – 1990)
- Direct Interface to ARM's EmbeddedICE™

Refer to the *ARM7DI Data Sheet* for additional information, including:

- Programmer's Model
- Instruction Set
- Instruction Cycle Operation (also refer to 'Memory Cycles Calculation' in Chapter 5)
- JTAG and Debug Interface

BLOCK DIAGRAM

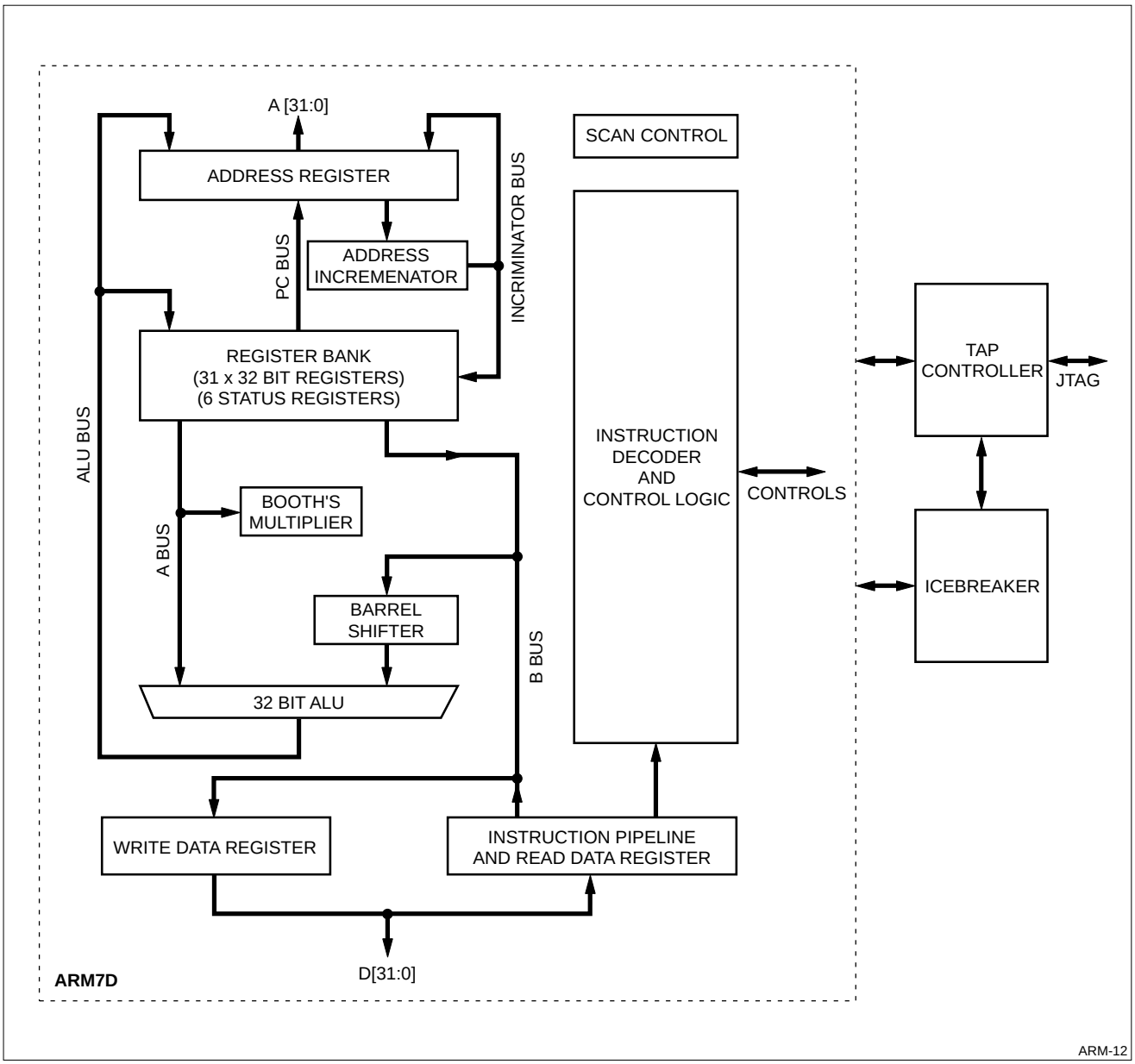


Figure 3-1. ARM7DI Core Block Diagram

Chapter 4

ON-CHIP MEMORY

INTRODUCTION

The 790A performance is maximized by its on-chip cache and local SRAM. The cache stores frequently used instructions and data by the ARM7DI core. Its zero-wait state, flexible modes of operation and write-back policy improves execution performance, reduces external bus traffic, and facilitate context switching. The local SRAM allows fast access to critical data or code such as interrupt service routines. It is relocateable and expandable.

CACHE

The 2K-Byte on-chip cache provides for zero wait state operation on cache hits. It is a combined data/instruction cache.

Features

- 2KB Cache
- 4-Way Set Associative
- Line Size = 1 Word
- 128 Sets
- Combined Instruction/Data
- Write-Back Policy
- Least Recently Used Replacement Policy
- Four Modes of Operation:
  - Cache Mode
  - SRAM Mode
  - Flush Mode
  - Invalidate Mode

Register Map

The base address for the Cache register(s) is 0x0FFFFA400. The offset, initial value, access and number of bits for each register are as follows:

Table 4-1.
Cache Register Offset

CACHE REGISTER	ADDRESS OFFSET [7:0]	RESET VALUE	ACCESS	NUMBER OF BITS
CCR	0x00	0x00	R/W	8

General Operation

The on-chip 2KB cache combines both instructions and data. It provides for zero wait state operation on cache hits and is optimized to minimize delays on cache misses. Memory updates are done via write-back to reduce traffic on the external bus. As a replacement policy, the cache uses the Least Recently Used algorithm (LRU). The cache can operate in four modes: Cache, SRAM, Flush, or Invalid mode.

Register Description

Cache Control Register (CCR)

7	6	5	4	3	2	1	0
///	///	///	///	I	F	S	E

The cache has an 8-bit control register to control its functionality. Upon reset, the cache is disabled.

Table 4-2.
CCR Fields

BIT POSITION	FIELD NAME	FUNCTION
0	E*	Cache Mode Bit 0: Cache Mode is Disabled 1: Cache Mode is Enabled
1	S*	SRAM Mode Bit 0: SRAM Mode is Disabled 1: SRAM Mode is Enabled
2	F*	Flush Mode Bit 0: Flush mode is Disabled 1: Flush mode is Enabled
3	I	Invalidate Mode Bit 0: Cache is Valid 1: Cache is Invalid
7:4	—	Reserved

NOTE: *Bits E, S, and F of CCR are mutually exclusive. Only one bit should be a logical '1' at a time.

Cache Mode

In Cache mode, the cache functions as a 512-word combined Instruction/Data cache. The cache is 4-way set associative with 128 sets. Each set consists of four 1-word lines of cached data (Data RAM), and four lines of 3-bit status and 23-bit address tag (Tag RAM) as shown in Figure 4-1. The cache uses a Write-Back policy. When a cache write occurs, the associated tag is marked modified. When that entry is later replaced, main memory gets updated. Write-Back reduces external memory traffic but leaves the main memory with old data until it is updated.

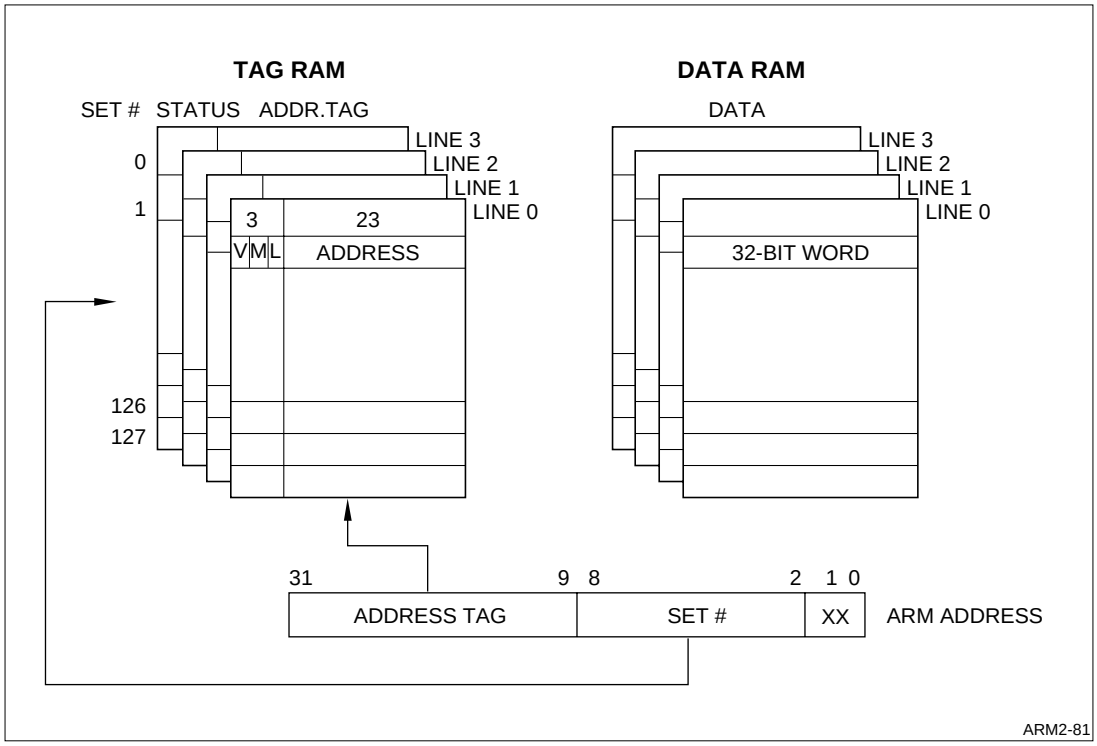


Figure 4-1. Cache Organization in Cache Mode

Each line has its own 3-bit status bit as shown in figure 4-1. Table 4-3 describes the function of each status bit.

During a cache-write back, the memory access to write the data back to cached memory will not be checked for system/user privilege violation because a check was already performed when the data was originally brought into the cache.

Table 4-3.
Cache Status Bits

STATUS BIT	NAME	DESCRIPTION
V	Valid Bit	0: The Tag is invalid and should be ignored 1: The Tag is valid
M	Modified Bit	0: Data is not modified 1: Data has been modified and has not yet been updated in external memory.
L	LRU Bit	Used for Least Recently Used Replacement Algorithm

SRAM Mode

In SRAM mode, the Data RAM part of the cache is accessed as SRAM (word access only). This will allow the system designer to examine the cached data and to use the cache as an extra on-chip SRAM. In SRAM mode, the Data RAM (512 words) will map to address locations 0x60000800-0x60000FFF and any access to this region will always access the cache's Data RAM.

In this mode, the system/user privileges for the Data RAM will be under the control of the Segment Descriptor Register (SDR) associated with its address or the Default Segment (SDR8) if no SDR is associated with its address (Refer to 'Memory and Peripheral Interface' and 'Memory Map and Register Summary' Chapters). Other bits in the SDR do not apply. All accesses are word accesses, byte accesses are not allowed.

The relationship between the mapped address and its content is shown in Figure 4-2 and Table 4-4. For example, address 0x60000E04 will contain the third line of the first set in the Data RAM.

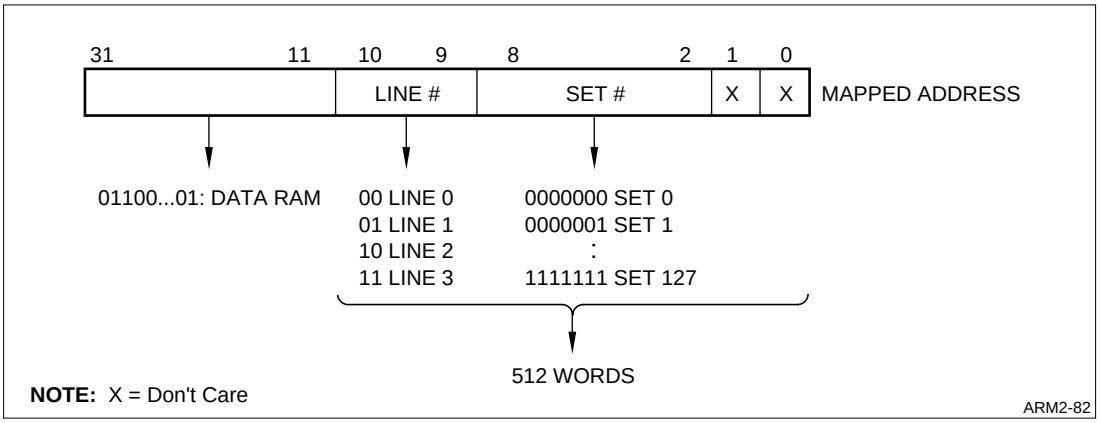


Figure 4-2. Cache Mapping in SRAM Mode

Table 4-4.
Cache Mapping in SRAM

MEMORY MAP ADDRESS	MEMORY CONTENT		
0x60000FFC	Data RAM	Set 127	Line 3
0x60000FF8	Data RAM	Set 126	Line 3
:	:	:	:
0x60000E04	Data RAM	Set 1	Line 3
0x60000E00	Data RAM	Set 0	Line 3
0x60000DFC	Data RAM	Set 127	Line 2
0x60000DF8	Data RAM	Set 126	Line 2
:	:	:	:
0x60000C04	Data RAM	Set 1	Line 2
0x60000C00	Data RAM	Set 0	Line 2
0x60000BFC	Data RAM	Set 127	Line 1
0x60000BF8	Data RAM	Set 126	Line 1
:	:	:	:
0x60000A04	Data RAM	Set 1	Line 1
0x60000A00	Data RAM	Set 0	Line 1
0x600009FC	Data RAM	Set 127	Line 0
0x600009F8	Data RAM	Set 126	Line 0
:	:	:	:
0x60000808	Data RAM	Set 2	Line 0
0x60000804	Data RAM	Set 1	Line 0
0x60000800	Data RAM	Set 0	Line 0

Flush Mode

In Flush mode, the cache will be treated as an SRAM (see SRAM mode) and an access to any cache set (0 - 127) that has any of the four lines modified (M = 1), will cause the modified line(s) to be written back to main memory. This feature allows the content of the cache to be written to main memory by accessing all 128 sets (0x60000800 – 0x600009FC) sequentially thus preserving data consistency.

Invalidate Mode

In Invalidate mode, all cache lines will be invalidated. This will force memory accesses to initially miss the cache and go to main memory. This feature is useful for context switching.

LOCAL SRAM

The 2K-Byte local SRAM provides zero wait state operation and allows fast access to critical data or code such as interrupt service routines.

Features

- 2K On-Chip SRAM
- Expandable to 4KB
- Programmable Read/Write
- Programmable User/System Privileges
- Relocateable

Register Map

The base address for the Local SRAM register(s) is 0xFFFFA404. The offset, initial value, access and number of bits for each register is as follows:

Table 4-5.
Local SRAM Register Offset

LOCAL SRAM REGISTER	ADDRESS OFFSET [7:0]	RESET VALUE	ACCESS	NUMBER OF BITS
LSCR	0x00	0x00	R/W	8

General Operation

The on-chip 2KB local SRAM provides for zero wait state operation allowing fast access to critical data or code. The 2KB local SRAM can be mapped into either low memory [0x00000000:0x000007FF], or high memory [0x60000000:0x600007FF]. For system/ user privileges, the local SRAM will be under the control of the Segment Descriptor Register (SDR) associated with its address or the Default Segment (SDR8) if no SDR is associated with its (Refer to 'Memory and Peripheral Interface' and 'Memory Map and Register Summary' Chapters). Other bits in the SDR do not apply. It can also be combined with the On-Chip cache for a total of 4KB of local SRAM as has been described in the cache section. Local SRAM is not cacheable.

Register Description

Local SRAM Control Register (LSCR)

7	6	5	4	3	2	1	0
///	///	///	///	///	///	L	E

The local SRAM has an 8-bit control register to control its functionality. Upon reset, the local SRAM is disabled.

Table 4-6.
LSCR Fields

BIT POSITION	FIELD NAME	FUNCTION
0	E	Local SRAM Enable Bit 0: Local SRAM is Disabled 1: Local SRAM is Enabled
1	L	Local SRAM Location 0: Local SRAM Mapped into low memory at: [0x00000000:0x000007FF] Any access to this region with Local SRAM enabled, will always access Local SRAM 1: Local SRAM Mapped into high memory at: [0x60000000:0x600007FF] Any access to this region with Local SRAM enabled, will always access Local SRAM
7:2	–	Reserved

Chapter 5

MEMORY AND PERIPHERAL INTERFACE

INTRODUCTION

The 790A supports standard x8 and x16 SRAM, DRAM, EEPROM, and Flash memory devices as well as memory mapped peripherals through a programmable external bus controller with little or no glue logic. This results in lower power consumption and savings in cost and board area.

FEATURES

- Supports 26-bit Address Bus
- Supports 16-bit Data Bus
- SRAM Controller
  - 6 SRAM Banks
  - Supports 64M Byte Bank
- DRAM Controllers
  - 2 DRAM Banks
  - Supports 128M Byte Bank
- Programmable Properties:
  - 6 Multiplexed Chip Enables/$\overline{CAS}$ Pins
  - 2 RAS Pins
  - 8 Wait States
  - 8/16-bit Bus Width
  - Half-Word Access
  - Cacheable
  - Access Privilege for User/System
  - DRAM Properties:
    - Page Mode
    - Bank Size
    - CAS Before RAS Refresh
    - Refresh Rate
- External Memory Mapped I/O Support
- LCD Frame Buffer Support

BLOCK DIAGRAM

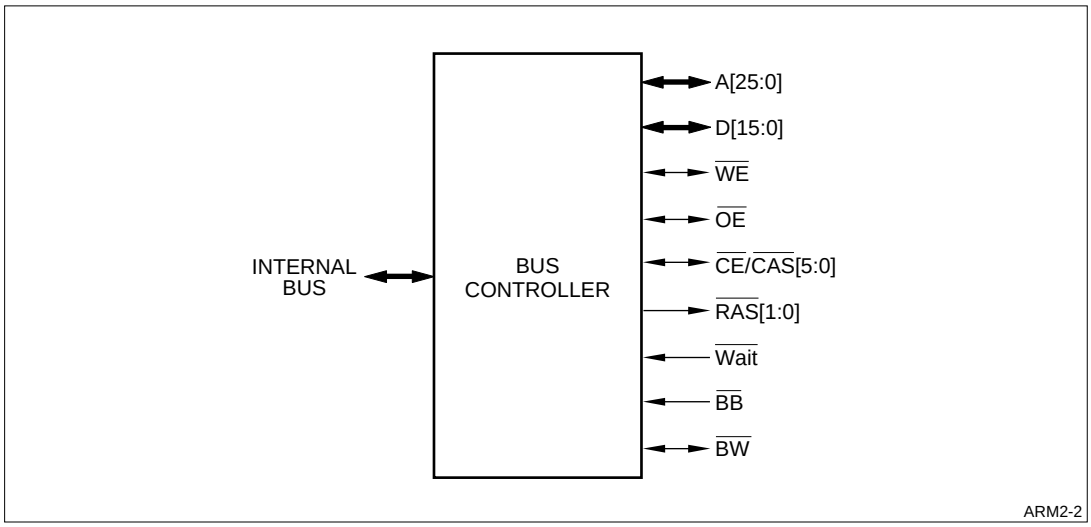


Figure 5-1. Memory and Peripheral Interface Block Diagram

REGISTER MAP

The base address for the Memory Segment Registers is 0xFFFFA000. The offset, initial value, access and number of bits for each register is as follows.

Table 5-1.
Memory and Peripheral Interface Register Offset:
Memory Segment Registers

REGISTER	ADDRESS OFFSET	RESET VALUE	ACCESS	NUMBER OF BITS
START0	0x00	0x0000	R/W	32
START1	0x04	0x0000	R/W	32
START2	0x08	0x0000	R/W	32
START3	0x0C	0x0000	R/W	32
START4	0x10	0x0000	R/W	32
START5	0x14	0x0000	R/W	32
START6	0x18	0x0000	R/W	32
START7	0x1C	0x0000	R/W	32

Table 5-2.
Memory and Peripheral Interface Register Offset:
Memory Segment Registers (cont'd)

REGISTER	ADDRESS OFFSET	RESET VALUE	ACCESS	NUMBER OF BITS
STOP0	0x20	0x0000	R/W	32
STOP1	0x24	0x0000	R/W	32
STOP2	0x28	0x0000	R/W	32
STOP3	0x2C	0x0000	R/W	32
STOP4	0x30	0x0000	R/W	32
STOP5	0x34	0x0000	R/W	32
STOP6	0x38	0x0000	R/W	32
STOP7	0x3C	0x0000	R/W	32
SDR0	0x40	0x0000	R/W	16
SDR1	0x44	0x0000	R/W	16
SDR2	0x48	0x0000	R/W	16
SDR3	0x4C	0x0000	R/W	16
SDR4	0x50	0x0000	R/W	16
SDR5	0x54	0x0000	R/W	16
SDR6	0x58	0x0000	R/W	16
SDR7	0x5C	0x0000	R/W	16
SDR8[1]	0x60	0x7801	R/W	16

NOTE: See 'Upon Reset' section.

The base address for the Bank Configuration Registers is 0xFFFFA100. The offset, initial value, access and number of bits for each register is as follows.

Table 5-3.
Memory and Peripheral Interface Register Offset:
Bank Configuration Register

REGISTER	ADDRESS OFFSET	RESET VALUE	ACCESS	NUMBER OF BITS
BCR0[1]	0x00	0x7003 ($\overline{BB}$= 0) 0xF009 ($\overline{BB}$ = 1)	R/W	16
BCR1	0x04	0x0000	R/W	16
BCR2	0x08	0x0000	R/W	16
BCR3	0x0C	0x0000	R/W	16
BCR4	0x10	0x0000	R/W	16
BCR5	0x14	0x0000	R/W	16
BCR6a	0x18	0x0000	R/W	16
BCR7a	0x1C	0x0000	R/W	16
BCR6b	0x20	0x0000	R/W	8
BCR7b	0x24	0x0000	R/W	8
DRR	0x28	0x0000	R/W	16

NOTE: See 'Upon Reset' section.

UPON RESET

Upon reset, the 790A uses $\overline{BB}$, the default segment (SDR8), and bank0 register (BCR0) to access the external boot memory. $\overline{BB}$ is used as a select signal. If $\overline{BB}$ is LOW, the default bank (BCR0) will be initialized to the value 0x7003 to allow the 790A to interface with a x8 boot memory. If $\overline{BB}$ is HIGH, the default bank (BCR0) will be initialized to the value 0xF009 to allow the 790A to interface with a x16 boot memory. The state of $\overline{BB}$ is sampled and captured on the rising edge of $\overline{RESETI}$. It is recommended that $\overline{BB}$ be permanently tied LOW for x8 boot memory or HIGH for x16 boot memory.

GENERAL OPERATION

The Bus Controller is responsible for controlling all external activities such as:

- Memory Interface
- Peripheral Interface
- DRAM Refresh Requests (Priority 1)
- LCD Controller Requests (Priority 2)
- Cache External Requests (Priority 3)
- ARM7DI External Requests (Priority 4)

The bus controller arbitrates between the various memory requests according to their priorities. The controller treats external peripherals as memory mapped I/O. The controller supports an external 26-bit address bus and 16-bit data bus. Six multiplexed chip selects/$\overline{\text{CAS}}$ lines are provided to support 6 SRAM banks, or 4 SRAM banks and 2 DRAM banks. Each bank has programmable properties to allow interfacing to a wide range of external memory and peripheral devices. The controller takes care of data alignment between the 16-bit external data bus and the 32-bit internal data bus and preserves compatibility with the ARM. The external bus controller also manages the logical memory space and maps it to the physical memory space through a memory management scheme.

MEMORY MANAGEMENT

The ARM7DI can address up to 4GB of address space (32-bit internal address). The 790A can address up to 64MB externally (26-bit external address). The 790A supports up to eight programmable segments (0 - 7), a default segment, and six programmable chip enables. Each segment/chip enable pair can address 64MB of external SRAM or 128MB of external DRAM. This will give a total of 384MB (6 x 64MB) external SRAM addressable space or in the case where two of the segments are DRAMs, the total memory addressable space is 512MB (4 x 64MB SRAM + 2 x 128MB DRAM).

Each of the eight segments consist of 3 registers, a Segment Descriptor Register (SDR), a START register and a STOP register. The SDR contains information about the system/user privileges, cacheability, half-word mode, and bank selection. The START and STOP registers determine the boundaries of the segment. Upon reset, all of the above 3 registers will be initialized to all 0's for all 8 segments. A Default Segment, segment 8, is used after reset. The default segment has its own SDR, SDR8, and spans the whole address space (0: 2^{32} - 1) and it uses Bank0 as the default bank. The default segment is used when the memory address does not match any of the eight main segments. The upper 64KB of the memory map is reserved for system configuration registers and internal peripheral registers (see Memory Map chapter).

The embedded controller supports eight banks, six banks for SRAM and two banks for DRAM. A SRAM bank will reflect the properties of an external SRAM (or SRAM like devices) such as chip selects, wait count, and bus width. A DRAM bank will reflect the properties of an external DRAM such as $\overline{\text{CAS}}$, $\overline{\text{RAS}}$, bank size, page mode, wait count, and bus width.

Logical To Physical Mapping

When the bus controller receives an internal address for a memory access, it maps the address to the appropriate segment (or default segment) and performs all the necessary checks for cacheability and privileges as programmed in the SDR. Once all the checks are passed, the bus controller accesses the appropriate memory bank control register, BCR, as programmed in the SDR. The BCR tells the bus controller which external device to select and its properties. The lower 26 bits of the address are used to address the external device.

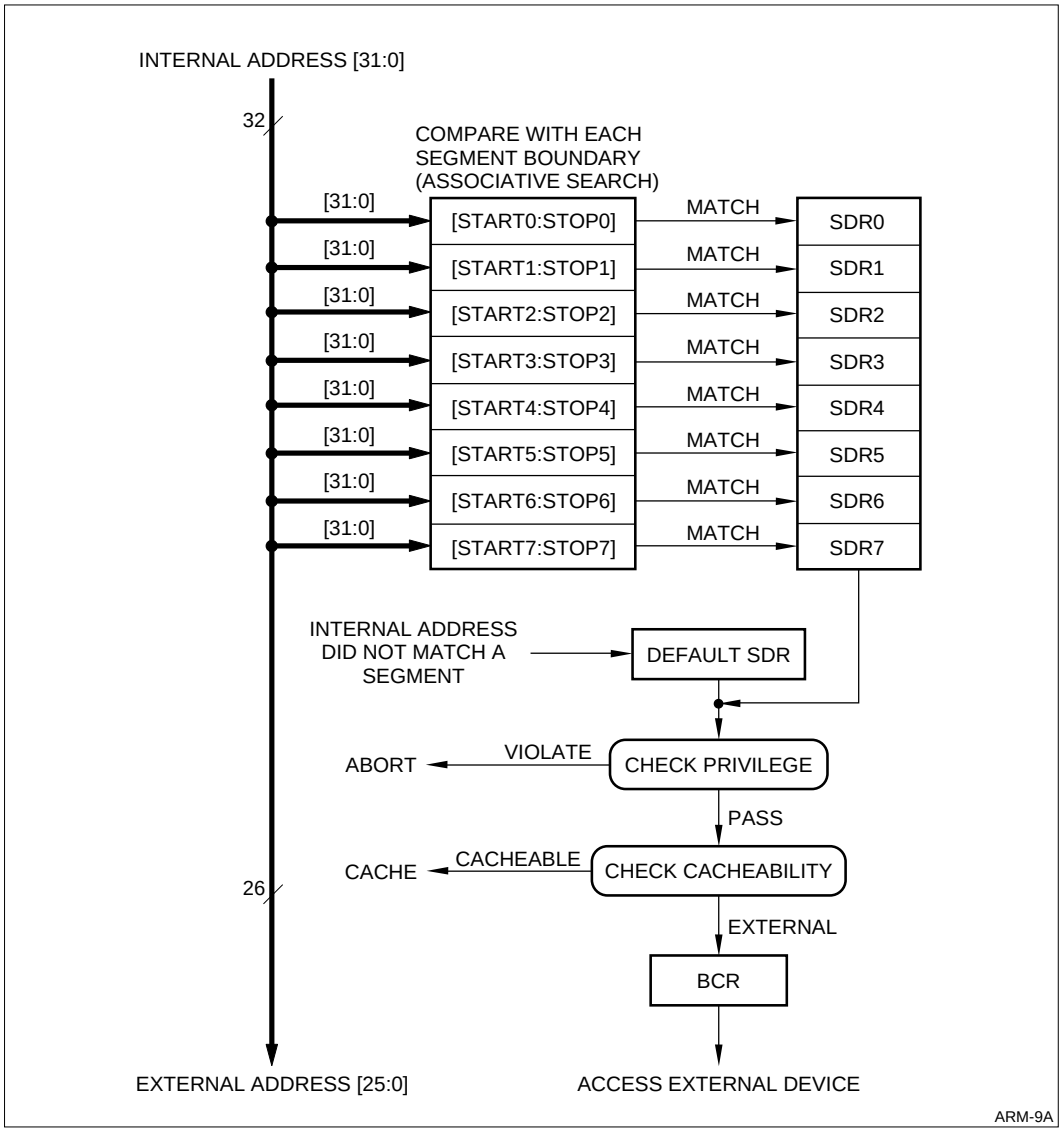


Figure 5-2. Logical to Physical Address Mapping

Memory Segments

The 790A supports up to eight programmable segments and a default segment. The user has the flexibility of defining the boundaries of each segment. *Segments should NOT overlap, otherwise UNPREDICTABLE operations will result.* Each segment has three registers associated with it:

1. Segment Descriptor Register (SDR)
2. START Register (START)
3. STOP Register (STOP)

The default segment has one register associated with it, SDR8.

Table 5-5.
SDR, START and STOP Association

SEGMENT	SDR	START	STOP
0	SDR0	START0	STOP0
1	SDR1	START1	STOP1
2	SDR2	START2	STOP2
3	SDR3	START3	STOP3
4	SDR4	START4	STOP4
5	SDR5	START5	STOP5
6	SDR6	START6	STOP6
7	SDR7	START7	STOP7
8 (Default)	SDR8	-	-

The user should also keep in mind that even though the internal bus supports up to 4GB of addressable space, the external bus supports only 64MB of addressable SRAM space per segment. If a given segment is larger than 64MB, the external address will wrap around beyond 64MB. For DRAM, the external bus supports 128MB of addressable space per segment.

Segments allow the user to map two or more memory regions the same physical external memory but with different privileges thus allowing for memory protection. The 790A supports four forms of privileges: user read, user write, system read, and system write. Since the ARM7DI supports multiple modes of operations (User, Supervisor, FIQ, IRQ, Abort, and Undefined), the definition of system mode is any mode other than User mode.

The Segment Descriptor Register (SDR) allows the user to program and control the privileges, cacheability, 16/32-bit mode, and bank selection.

The Default Segment, segment 8, is defined to allow access to external memory upon Reset and is used as a default segment if a memory address does not belong to segments 0 - 7. This segment has its own SDR, SDR8, and spans the full address space ($0:2^{32} - 1$). The default segment points to Bank0 as the default bank to use.

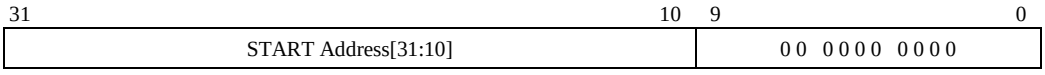
Memory Banks

The 790A supports up to six SRAM banks and two DRAM banks. The banks reflect the properties of the external device and controls the behavior of the external bus controller.

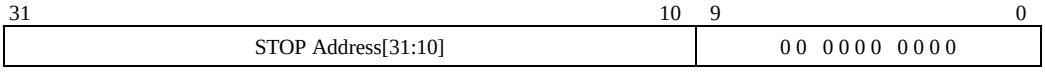
REGISTER DESCRIPTION

START and STOP Registers (STARTn, STOPn)

START Register



STOP Register



There are eight pairs of START/STOP registers, one for each segment (see Table 5-5a). The default segment does not have a START/STOP pair because it spans the whole address space. START points to the beginning of a segment and STOP points to the end of a segment plus one. Segments must be an integer multiple of 1KB pages. A memory address belongs to a segment if:

START[31:10] ≤ Memory Address [31:10] < STOP[31:10]

When the bus controller receives an address, it compares the address against all eight segments. If a match is found, the corresponding SDR is selected. If no match was found, the default SDR is selected. Segments should **NOT** overlap.

Since segments must be an integer multiple of 1KB pages, only the high order 22 bits of the address are needed. A write access to START/STOP registers will write 0s into bits [9:0]. Upon Reset, START and STOP registers will initialize to all 0s forcing any external memory access to go to the default segment.

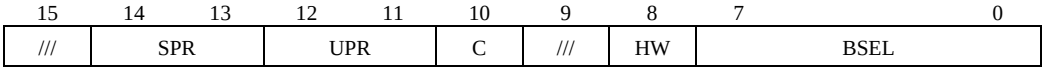
Example:

To assign a 2KB region to segment 2 starting at address 0x00000400 and ending at address 0x00000BFF, the following assignment will define the region:

START2 <== 0x00000400

STOP2 <== 0x00000C00 (0x00000BFF+1)

Segment Descriptor Registers (SDRn)



There are nine Segment Descriptor Registers, SDR0 - SDR8 (see Table 5-5a). The user can program and control the privileges, cacheability, 16/32-bit mode, and bank selection for each segment.

Table 5-6.
SDR Fields

BIT POSITION	FIELD NAME	FUNCTION
15	///	Reserved
14:13	SPR	System Privilege 00: No privileges 01: Read only privilege 10: Write only privilege 11: Read/Write privilege
12:11	UPR	User Privilege 00: No privileges 01: Read only privilege 10: Write only privilege 11: Read/Write privilege
10	C	Cacheability 0: Segment is not cacheable 1: Segment is cacheable
9	///	Reserved
8	HW	Half-Word Mode 0: 32-bit Mode 1: 16-bit Mode
7:0	BSEL	Bank Select 0000 0001: Select Bank0 0000 0010: Select Bank1 0000 0100: Select Bank2 0000 1000: Select Bank3 0001 0000: Select Bank4 0010 0000: Select Bank5 0100 0000: Select Bank6 1000 0000: Select Bank7 Any other combination is *illegal.*

System/User Privileges

Each segment can be individually programmed to give the user and/or the system read and/or write access to the memory space associated with the segment. This will allow for memory management and protection. As an example, both code and data can reside in the same external SRAM with code accessible by the system only and data by the system and the user. A violation of the programmed privileges will cause an ABORT (data or prefetch). Write accesses that are aborted are not performed and thus do not corrupt memory.

All memory accesses (Cacheable, Non-cacheable, Local SRAM...) generated by the LH77790A core will be checked for privilege violation with the exception of two operations, cache write-back and LCD refresh accesses. During a cache write-back, the memory access to write the data back to cached memory will not be checked because a check was performed when the data was originally brought into the cache.

LCD Controller refresh requests are treated as system level requests. However, LCD requests will not be checked for any privilege violation.

Cacheability

A segment can be cacheable in which case any access to that segment will be handled by the cache or non-cacheable in which case the access will be handled by the external bus controller. Since the frame buffer for the LCD is located in main memory, the segment containing the frame buffer should not be cacheable. The external bus controller will ignore the cacheability bit when it receives an LCD request.

Half-Word Mode (HW)

The ARM7DI core supports only Byte (8-bits) and Word (32-bits) memory accesses. Since the 790A has a 16-bit data bus, a word access will take two memory accesses. In HW mode, the 790A allows Half-Word (HW) accesses to the external memory. This means that the 790A will treat all word accesses (read or write) as HW accesses and will take one memory access. On a word read, the 16-bit data from memory will be sign extended in the external bus controller and sent to the ARM7DI as a 32-bit word. On a word write, the external bus controller will write the low order 16-bits of the word from the ARM7DI to external memory.

As mentioned above, the ARM7DI is unaware of HW mode. All LOAD and STORE instructions are treated as byte or word instructions. For example a LDM instruction will update the destination address by 4 regardless of the state of HW bit. So LDM and STM will not operate correctly in HW mode.

Cache does not recognize HW mode. Any memory segment that is operating in HW mode should be programmed to be non-cacheable to avoid data corruption.

Local SRAM does not recognize HW mode. Local SRAM segment should be programmed to operate in 32-bit mode (HW = 0).

This mode is very important when it comes to programming x16 Flash memories. Programming a x16 Flash memory requires accessing the Flash in a certain sequence that tells the Flash memory it is getting a command on the data bus instead of data. To program such a Flash, HW mode should be used.

Bank Selection

Bank Selection field consists of eight mutually exclusive bits. A '1' in bit position i selects Banki and a '0' in all positions will select Bank8. All valid combinations are listed in table 5-6. *Using any other value will cause UNPREDICTABLE results.*

Reset Value

Upon Reset, SDR0-7 will be initialized to 0x0000 and SDR8 will be initialized to 0x7801 (system read/write privileges, non-cacheable segment, 32-bit mode, Bank0 select).

Bank Control Registers (BCRn)

The 790A supports up to nine banks of external memory and peripherals. Each bank has a Bank Control Register, BCR, associated with it that allows the user to program the properties of the external device it is connected to and will control the behavior of the external bus controller. Banks(0 - 5) are SRAM/ROM/... Banks and Banks(6 - 7) are DRAM banks. BCR0-5 allow the user to program the external chip enables, wait cycles, and bus width. BCR6 has two registers, BCR6a which allows the user to program CAS lines, wait cycles, and bus width; BCR6b which has DRAM bank size and page mode. BCR7 is identical to BCR6. BCR6 controls $\overline{RAS0}$ and BCR7 controls $\overline{RAS1}$.

BCR0-5

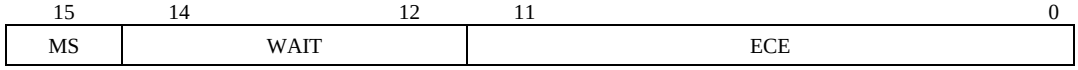


Table 5-7.
BCR0-5 Fields

BIT POSITION	FIELD NAME	FUNCTION
15	MS	External SRAM Bus Size 0: ×8 SRAM connected to data bus 1: ×16 SRAM connected to data bus
14:12	WAIT	Wait Cycles for external SRAM 000: 0 Wait cycles (for fast SRAM) 001: 1 Wait cycle 010: 2 Wait cycles : 111: 7 Wait cycles (for slow SRAM)
11:00	ECE	External Chip Enable Controls. Every Chip Enable pin ($\overline{CE0}$ - $\overline{CE5}$) has two control bits to allow maximum flexibility for connecting to x8 and x16 SRAMs.

MS

This bit tells the external bus controller the bus size of the external SRAM in order to produce the correct number of memory transfers as shown below.

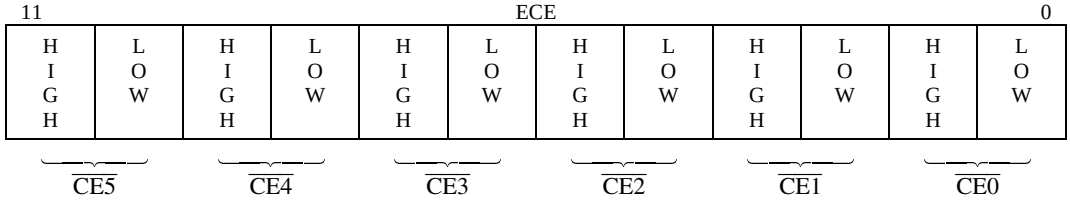
Table 5-8.
External SRAM Width

ACTION	MS	EXTERNAL SRAM WIDTH	NUMBER OF MEMORY TRANSFER
Read 32-bits	0	8	4
Read 32-bits	1	16	2
Read 16-bits	0	8	2
Read 16-bits	1	16	1
Read 8-bits	0	8	1
Read 8-bits	1	16	1

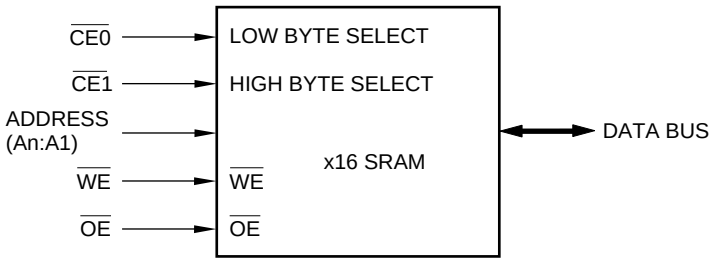
WAIT

This field allows SRAMs with various speeds to be connected to the 790A with no extra external logic.

ECE



These 12 bits control the six external $\overline{CE}$/$\overline{CAS}$ pins. For SRAM, these pins will be used as $\overline{CE}$ pins. Each pin is controlled by 2 bits from ECE field to support byte enables on x16 SRAMs as shown in Figure 5-3. Each $\overline{CE}$ pin can be enabled for low byte, high byte, or both depending on address bit0 (A0), external memory bus size (MS) and access type (Byte, Half-Word, or Word Loads and Stores) as shown in Table 5-11.



NOTES:
1. CE0 is the LOW byte chip select.
2. CE1 is the HIGH byte chip select.
3. Program ECE field as follows: 00 00 00 00 10 01
   - For Byte Read/Write, CE0 will be active if A0 = 0, CE1 will be active if A0 = 1.
   - For Non-Byte Read/Write, CE0 and CE1 will both be active regardless of A0.

ARM2-4

Figure 5-3. ECE Example

BCR6a and BCR7a

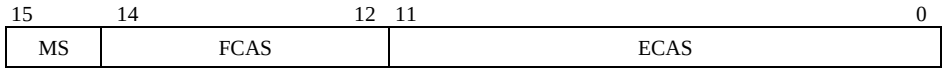


Table 5-9.
BCR6a-BCR7a Fields

BIT POSITION	FIELD NAME	DESCRIPTION
15	MS	External DRAM Bus Size 0: ×8 DRAM connected to data bus 1: ×16 DRAM connected to data bus
14:12	FCAS	14:12 FCAS First Transfer CAS Width. It also programs RAS, width and Precharge time as follows: FCAS — First CAS — RAS — Precharge Time 000 — illegal — illegal — illegal 001 — 1.5 cycles — 2 cycles — 2 cycles 010 — 2.5 cycles — 3 cycles — 3 cycles : — : — : — : 111 — 7.5 cycles — 8 cycles — 8 cycles RAS to CAS delay is always 1 cycle.
11:0	ECAS	External CAS Controls. Every CAS pin (CAS0 - CAS5) has two control bits to allow maximum flexibility for connecting to x8 and x16 DRAMs

NOTE: BCR6 controls RAS0 and BCR7 controls RAS1

MS

This bit tells the external bus controller the bus size of the external DRAM in order to produce the correct number of memory transfers as shown in Table 5-10.

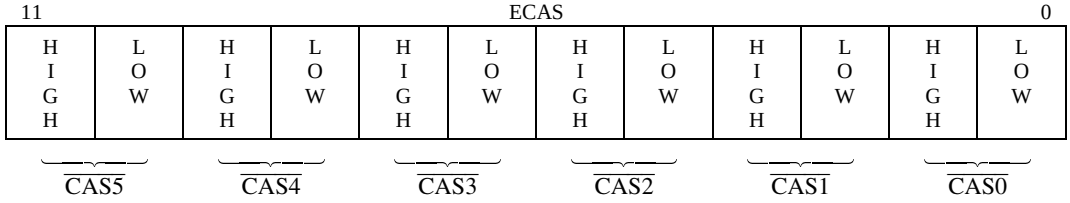
Table 5-10.
External DRAM Width

MEMORY ACCESS	MS	EXTERNAL DRAM WIDTH	NUMBER OF MEMORY TRANSFERS
Read 32 bits	0	8	4
Read 32 bits	1	16	2
Read 16 bits	0	8	2
Read 16 bits	1	16	1
Read 8 bits	0	8	1
Read 8 bits	1	16	1

FCAS

In DRAM normal mode access, each transfer will consist of a RAS, CAS, and precharge cycles of width FCAS. In DRAM Page mode, the first transfer will consist of an activating $\overline{RAS}$ followed by a CAS cycle of width FCAS. Subsequent accesses/transfers will have CAS cycles controlled by BCAS field in BCR6b and BCR7b. $\overline{RAS}$ remains active until a DRAM page boundary is crossed, as defined in Table 5-13, or refresh needs to be performed, $\overline{RAS}$ will deactivate and a precharge cycle of width FCAS cycles will be performed.

ECAS



These 12 bits control the six external $\overline{CE}/\overline{CAS}$ pins. For DRAM, these pins will be used as $\overline{CAS}$ pins. Since there are only two DRAM banks, a maximum of four $\overline{CAS}$ pins can be used. The other two pins are used as $\overline{CE}$. Each pin is controlled by 2-bits from the ECAS field to support byte enables in x16 DRAMs with two $\overline{CAS}$ byte controls. Each $\overline{CAS}$ pin can be enabled for low byte, high byte, or both depending on address bit0 (A0), external memory bus size (MS) and access type (Byte, Half-Word, or Word Loads and Stores) as shown in Table 5-11.

Table 5-11.
Activation of $\overline{CE}/\overline{CAS}$ and $\overline{BW}$

ACCESS	MS	ECE/ECAS (HIGH, LOW)	$\overline{CE}/\overline{CAS}$	$\overline{BW}$
Byte	X	(0, 0)	1	0
		(0, 1)	A0	0
		(1, 0)	$\overline{A0}$	0
		(1, 1)	0	0
Half-Word (HW bit is set to 1)	0	(0, 0)	1	1
		(0, 1)	A0	1
		(1, 0)	$\overline{A0}$	1
		(1, 1)	0	1
Half-Word (HW bit is set to 1)	1	(0, 0)	1	1
		(0, 1)	0	1
		(1, 0)	0	1
		(1, 1)	0	1
Word	0	(0, 0)	1	1
		(0, 1)	A0	1
		(1, 0)	$\overline{A0}$	1
		(1, 1)	0	1
Word	1	(0, 0)	1	1
		(0, 1)	0	1
		(1, 0)	0	1
		(1, 1)	0	1

NOTES:

1. Instruction fetches are always word accesses regardless of the external memory/peripheral bus size.
2. $\overline{BW}$ is not a function of the external memory/peripheral bus size. It is only active if the ARM7DI core is executing a load/store byte instructions.
3. $\overline{BW}$ is valid during external memory cycles. $\overline{BW}$ is a Don't Care during DRAM Refresh.
4. A typical usage of BW is to allow the use of one $\overline{CE}$ to access a x16 external device that is byte addressable. $\overline{BW}$, $\overline{CE}$, A0 can be externally decoded to access a low, high, or both bytes.
5. X = Don't Care

BCR6b and BCR7b

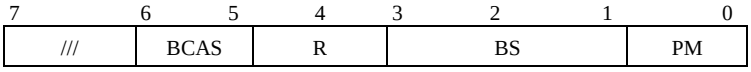


Table 5-12.
BCR6b - BCR7b Fields

BIT POSITION	FIELD NAME	FUNCTION
7	///	Reserved
6:5	BCAS	Burst Transfer $\overline{CAS}$ Width 00: 1.5 cycles 01: 1.5 cycles 10: 2.5 cycles 11: 3.5 cycles
4	R	Refresh Active 0: Refresh is not active 1: Refresh is active (assuming DRR ≠ 0) Refresh is active for both DRAM banks if BCR6b(4)=1 OR BCR7b(4)=1
3:1	BS	DRAM Bank Size 000: 256K x 8, 512K x 8, 256K x 16, 512K x 16 001: 1M x 8, 2M x 8, 1M x 16, 2M x 16 010: 4M x 8, 8M x 8, 4M x 16, 8M x 16 011: 16M x 8, 32M x 8, 16M x 16, 32M x 16 100: 64M x 8, 128M x 8, 64M x 16 Other combinations are invalid
0	PM	DRAM Page Mode 0: Normal Mode 1: Page Mode

DRAM Refresh Register (DRR)

'CAS before RAS', CBR, refresh cycles are performed periodically as determined by the DRAM Refresh Register (DRR). During CBR refresh, the width of $\overline{RAS0}/\overline{RAS1}$ is the same and is equal to the largest of the two FCAS values programmed in BCR6a and BCR7a. The CAS cycle starts one XCLK cycle before the RAS cycle and remains active for 3.5 cycles. The DRR is a 16-bit wide register. A ZERO programmed in the DRR will disable DRAM refresh. Low refresh values are not practical because CBR takes at least five cycles and the 790A will be constantly performing CBR. The logical OR of [BCR6b(4),BCR7b(4)] activates refresh for both Banks. Since both banks share the same DRR register, DRR should be programmed to accommodate DRAMs connected to both banks. Upon Reset, this register is initialized to all 0's (refresh disabled).

To properly disable DRAM refresh, the following sequence must be followed:

1. Program DRR to all 0's.

2. Program BCR6b(4) and BCR7b(4) to 0.

NOTES:

1. During Refresh, $\overline{CAS}$, $\overline{RAS}$ and $\overline{WE}$ are valid (Figure 5-15). $\overline{WE}$ is HIGH during refresh.
2. $\overline{CAS}$ before $\overline{RAS}$ (CBR) refresh and DRAM accesses are mutually exclusive. CBR will take place either before or after a DRAM access.
3. $\overline{CAS}$ before $\overline{RAS}$ (CBR) refresh and non-DRAM accesses can overlap. Non-DRAM accesses will be stretched while refresh is taking place and will complete after refresh is done. If a refresh takes place during a non-DRAM write access, $\overline{WE}$ will change from LOW state to HIGH state during refresh then back to LOW state after refresh to complete the write access.
4. When software sets SWRST register to '1' (see RESET Chapter) to assert $\overline{RESETO}$, DRAM refresh will stop. DRAM refresh can't be activated while SWRST register is set to '1'. When SWRST is set to '0', DRAM Refresh Register (DRR) must be written again with the refresh value to activate refresh.

The DRAM refresh rate relates to DRR as follows:

$$\text{Refresh Rate} = \text{DRR} * t_{XCLK}$$

Example 1

DRAM1 has a maximum refresh rate of 15.6 µs (1024 rows, 16 ms) and DRAM2 has a maximum refresh rate of 125 µs (1024 rows, 128 ms) according to the DRAM data sheet. The 790A is clocked with a 25 MHz clock (t_{XCLK} = 40 ns). DRR should be programmed to be less than 390 (15.6 µs/40 ns) to accommodate both DRAMs in the system.

NOTES:

1. t_{XCLK} is the system input clock period. t_{XCLK} is not affected by any internal division in the Power Management Unit.
2. To guarantee proper operation, the DRR should be programmed to be < $\overline{RAS}$ (max.) as specified by the DRAM manufacturer. This will guarantee that a refresh cycle will occur before $\overline{RAS}$ exceeds its pulse width maximum limit. This becomes an issue in two situations:
   a. 790A is running at a very small frequency (e.g. 80 KHz)
   b. 790A is running in page mode since $\overline{RAS}$ remains active after the first access until a page boundary is crossed or a fresh needs to be performed.

Example 2

If only DRAM2 (as defined in example 1) was used in the system, and DRAM2 has a max. RAS width of 100 µs. DRR should be programmed to be less than 2500 (100 µs/ 40 ns) as opposed to less than 3125 (125 µs/40 ns) to avoid violating the max. RAS width in page mode.

DRAM CONTROLLER

DRAM Access Mode

There are two modes of operations:

Normal Mode

In normal mode, a precharge cycle ends each DRAM access. Each DRAM access will require a new $\overline{RAS}$ cycle, $\overline{CAS}$ cycle(s) and a precharge cycle.

Page Mode

In page mode, $\overline{RAS}$ remains active until a page boundary is crossed or a refresh cycle needs to be performed. In page mode, the first DRAM transfer requires a $\overline{RAS}$ cycle. Subsequent DRAM transfers/accesses only require short $\overline{CAS}$ cycles thus reducing power consumption and access time for a sequence of memory accesses. In the 790A, both $\overline{RAS}$ signals can be active at the same time.

DRAM Page Size

Page size depends on the Bank Size (BS) programmed in BCR6b and BCR7b.

Table 5-13.
DRAM Page Size

BS FIELD	PAGE SIZE (BYTES)
000	512
001	1K
010	2K
011	4K
100	8K

DRAM Address Bus Multiplex

Addresses are presented to DRAM in two sequential transfers. The row address is presented with the $\overline{RAS}$ strobe, then the column address with the $\overline{CAS}$ strobe. The 790A will multiplex the internal address bits a[26:0] onto the external address bits A[25:0] as necessary.

During $\overline{CAS}$ cycles, no multiplexing is needed and internal address bits a[25:0] are put on the external address bus A[25:0]. During $\overline{RAS}$ cycles, the proper internal address bits are multiplexed on the external address bus as a function of the DRAM bank size programmed in BCR6b/BCR7b.

Address multiplexing in $\overline{RAS}$ cycles is independent of DRAM bus width (x8 or x16), but the way the DRAM is connected to the 790A depends on the DRAM bus width. The following two tables show how a x8 DRAM or x16 DRAM should be connected to the 790A. There is one table for x8 DRAMs, and another for x16 DRAMs. The tables show the external address pins of the 790A, A[25:0], the DRAM pins B[13:0], and the internal address bit a[26:0] mapped to the external address bus in $\overline{RAS}$ and $\overline{CAS}$ cycles. The tables also show the physical connections between the external address pins and DRAM address pins.

Table 5-14.
Address Bus Multiplex for x8 DRAMs

PHYSICAL CONNECTION		INTERNAL ADDRESS MULTIPLEX DURING $\overline{CAS}$	INTERNAL ADDRESS MULTIPLEX DURING $\overline{RAS}$*				
DRAM ADDRESS PIN NO. B[13:0]	790A EXTERNAL ADDRESS PIN NO. A[25:0]		256K x 8 512K x 8	1M x 8 2M x 8	4M x 8 8M x 8	16M x 8 32M x 8	64M x 8 128M x 8
B0	A0	a0	a9	a10	a11	a12	a13
B1	A1	a1	a10	a11	a12	a13	a14
B2	A2	a2	a11	a12	a13	a14	a15
B3	A3	a3	a12	a13	a14	a15	a16
B4	A4	a4	a13	a14	a15	a16	a17
B5	A5	a5	a14	a15	a16	a17	a18
B6	A6	a6	a15	a16	a17	a18	a19
B7	A7	a7	a16	a17	a18	a19	a20
B8	A8	a8	a17	a18	a19	a20	a21
B9	A9	a9	a18	a19	a20	a21	a22
B10	A10	a10	a19	a20	a21	a22	a23
B11	A11	a11	a11	a21	a22	a23	a24
B12	A12	a12	a12	a12	a23	a24	a25
B13	A13	a13	a13	a13	a13	a25	a26
	A14	a14	a14	a14	a14	a14	a14
	A15	a15	a15	a15	a15	a15	a15
	:	:	:	:	:	:	:
	A20	a20	a20	a20	a20	a20	a20
	:	:	:	:	:	:	:
	A25	a25	a25	a25	a25	a25	a25

NOTE: *Based on DRAM Bank Size programmed in BCR6b/BCR7b

Table 5-15.
Address Bus Multiplex for x16 DRAMs

PHYSICAL CONNECTION		INTERNAL ADDRESS MULTIPLEX DURING $\overline{CAS}$	INTERNAL ADDRESS MULTIPLEX DURING $\overline{RAS}$*				
DRAM ADDRESS PIN NO. [B13:0]	790A EXTERNAL ADDRESS PIN NO. [A25:0]		256K x 16 512K x 16	1M x 16 2M x 16	4M x 16 8M x 16	16M x 16 32M x 16	64M x 16
	A0	a0	a0	a0	a0	a0	a0
B0	A1	a1	a10	a11	a12	a13	a14
B1	A2	a2	a11	a12	a13	a14	a15
B2	A3	a3	a12	a13	a14	a15	a16
B3	A4	a4	a13	a14	a15	a16	a17
B4	A5	a5	a14	a15	a16	a17	a18
B5	A6	a6	a15	a16	a17	a18	a19
B6	A7	a7	a16	a17	a18	a19	a20
B7	A8	a8	a17	a18	a19	a20	a21
B8	A9	a9	a18	a19	a20	a21	a22
B9	A10	a10	a19	a20	a21	a22	a23
B10	A11	a11	a11	a21	a22	a23	a24
B11	A12	a12	a12	a12	a23	a24	a25
B12	A13	a13	a13	a13	a13	a25	a26
	A14	a14	a14	a14	a14	a14	a14
	A15	a15	a15	a15	a15	a15	a15
	:	:	:	:	:	:	:
	A20	a20	a20	a20	a20	a20	a20
	:	:	:	:	:	:	:
	A25	a25	a25	a25	a25	a25	a25

NOTE: *Based on DRAM Bank Size programmed in BCR6b/BCR7b

MEMORY CYCLES CALCULATION

This section calculates the typical number of memory cycles for accessing the On-chip SRAM, External non-DRAM (SRAM/ROM/...) and DRAM (including instruction fetches). The 790A inserts an address setup cycle at the beginning of every non-sequential access. The memory cycles can be used in conjunction with ARM's instruction cycle times (refer to ARM7DI Data Sheet) to calculate the instruction cycle (times).

Definition

A sequential cycle is any transfer to or from an address that is either the same address in the preceding cycle or one word after the preceding address (ignoring bits 0 and 1 of the address). For example, LOAD/STORE multiple instructions (LDM/STM) start with a non-sequential cycle for the first register transfer but consecutive register transfers are sequential. Only the first transfer has an address setup cycle. Consecutive code fetches is another example of sequential cycles with the exception of the first code fetch which is non-sequential.

NOTE: In the following memory cycle calculations, an idle cycle at the end of each Load/Load Multiple instruction was not included in the cycle equations. This idle cycle is inserted by the ARM7DI core to perform an internal transfer of the loaded data to the destination register. This extra cycle must be included in any performance calculations.

On-chip (local) SRAM

The local SRAM is a 32-bit wide, zero wait state memory. It can be accessed as bytes (8-bits) or words (32-bits).

C = AS + RT
C: Number of Cycles to perform a memory access.
AS: Address Setup Cycle, 1 Cycle
RT: Number of Registers Transferred

Table 5-18.
Memory Cycles: On-Chip (Local) SRAM

ACCESS	SYNTAX	AS	RT	C
Load Byte	LDRB	1	1	2
Store Byte	STRB	1	1	2
Load Word	LDR	1	1	2
Store Word	STR	1	1	2
Load Multiple	LDM	1	1 - 16	2 - 17
Store Multiple	STM	1	1 - 16	2 - 17

External Memory Devices

External memory devices vary in speed and configuration. The 790A can be programmed to generate up to seven wait states to accommodate various speed memory devices. For devices that require more delays, $\overline{WAIT}$ allows them to insert more wait states in addition to what has been programmed. The 790A supports x8 and x16 external devices. All of the above affects the number of memory cycles per memory access.

Non-DRAM (SRAM/ROM/EPROM/Flash/...)

C = AS + [(W + 1) x T x RT]

C: Number of cycles to perform a memory access.
AS: Address setup cycle, 1 cycle
W: Number of wait cycles programmed via WAIT field in BCR(0-5), plus the number of extra wait cycles due to $\overline{WAIT}$ = 0.
T: Number of transfers needed per memory access (see table 5-19).
RT: Number of registers transferred:
1 for normal loads and store (LDR, LDRB, STR, STRB)
1 - 16 for load an store multiple (LDM, STM).

Table 5-19.
Number of Transfers Per Memory Access

MEMORY WIDTH	# BYTES TO LOAD OR STORE	T	AS
8	1	1	1
8	2	2	1
8	4	4	1
16	1	1	1
16	2	1	1
16	4	2	1

DRAM

Normal Mode

$$C = AS + [(RAS + CAS + Precharge) \times T] \times RT$$

C: Number of cycles to perform a memory access
AS: Address setup cycle, 1 cycle
RT: Number of registered transferred:
1 for normal loads and store (LDR, LDRB, STR, STRB)
1 - 16 for load an store multiple (LDM, STM).
RAS: Number of cycles row address is valid ($\overline{RAS}$ to $\overline{CAS}$ delay)
CAS: Number of cycles column address is valid
Precharge: $\overline{RAS}$ precharge time (an SRAM access can start during this time)
T: Number of transfers needed per memory access. (Refer to table 5-19)

Table 5-20.
Normal Mode DRAM Cycle Parameters

AS	RAS	CAS	PRECHARGE	T
1	1	FCAS* + ½	FCAS* - ½	Table 5-19

NOTE: *Decimal value (e.g. FCAS = 010 ⇒ FCAS = 2.5) as defined in BCR6/BCR7

Page Mode

Assuming no page crossing, the number of memory cycles depends on whether RAS is active or not.

If RAS is not active,

$$C = \underbrace{\mathbf{AS + RAS + First\_CAS + (Burst\_CAS) \times (T - 1)}}_{\mathbf{1st\ Register\ Transfer}} + \mathbf{[Burst\_CAS \times T] \times (RT - 1)}$$

If RAS is active ,

$$C = AS + [Burst\_CAS \times T] \times RT$$

C: Number of cycles to perform a memory access
AS: Address setup cycle, 1 cycle
RAS: Number of cycles row address is valid ($\overline{RAS}$ to $\overline{CAS}$ delay)
First_CAS: Number of cycles the first column address is valid
Burst _CAS: Number of cycles burst column address is valid
T: Number of transfers needed per memory access. (Refer to table 5-19)
RT: Number of registered transferred:
1 for normal loads and store (LDR, LDRB, STR, STRB)
1 - 16 for load an store multiple (LDM, STM).

5-21.
Page Mode DRAM Cycle Parameter

AS	RAS	FIRST_CAS	BURST_CAS	T
1	1	FCAS* + ½	BCAS* + ½	Table 5-19

NOTE: *Decimal values as defined in BCR6/BCR7.

MEMORY CYCLE DIAGRAMS

The following section presents cycle based diagrams for SRAM and DRAM accesses in various modes of operation. The following statements apply to all diagrams:

1. All diagrams assume a 32-bit mode (HW = 0) of operation, and non-sequential access.
2. An address setup cycle is *always* needed at the *beginning* of each non-sequential access.
3. $\overline{BW}$ is valid during external memory cycles. $\overline{BW}$ is not valid during DRAM refresh.
4. The cycle diagrams are logical diagram. Refer to "LH77790A ARM-Based Embedded Microcontroller Thermal and Electrical Specifications" for AC parameters.
5. Choice of $\overline{CE}$/$\overline{CAS}$ and $\overline{RAS}$ is arbitrary.

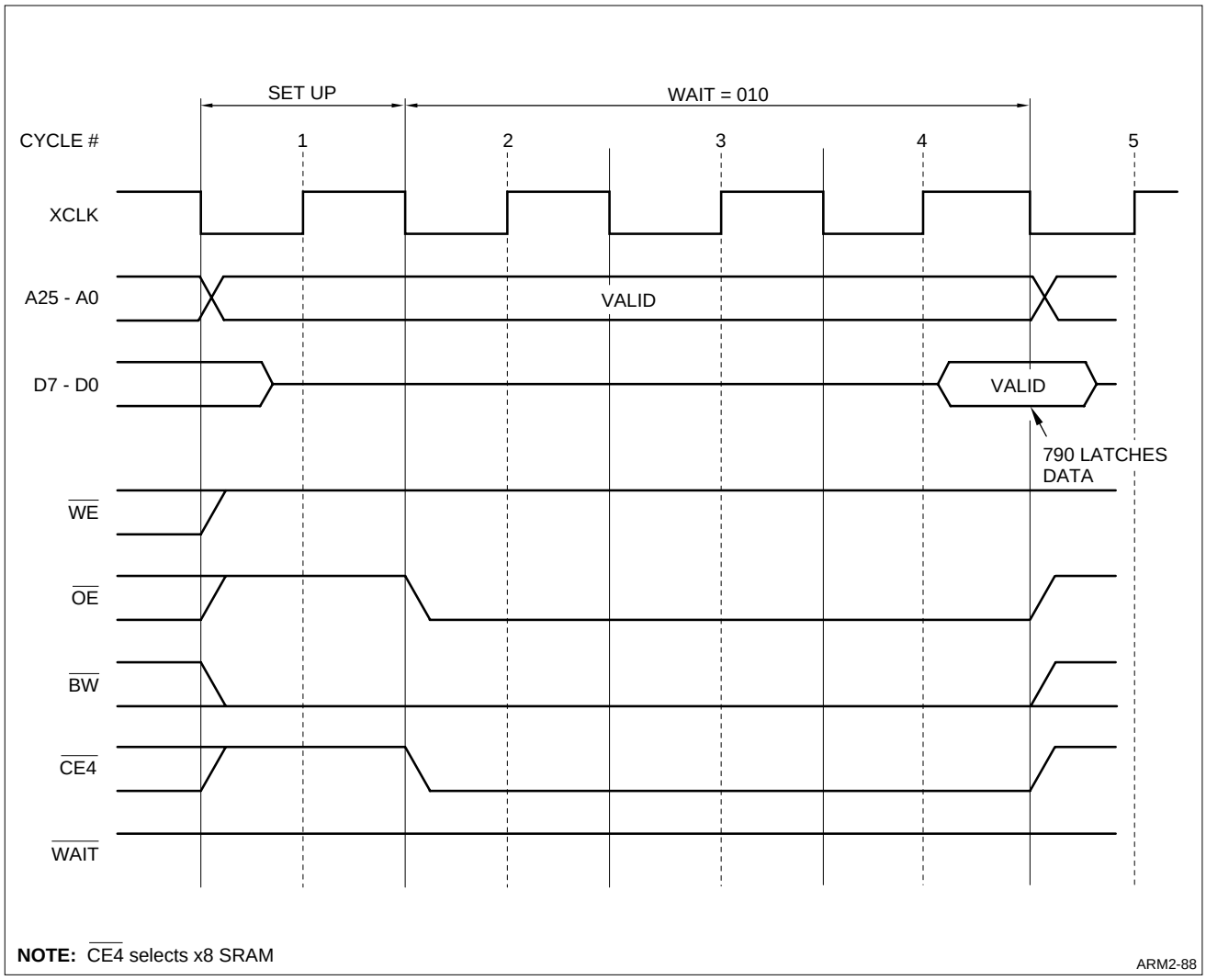
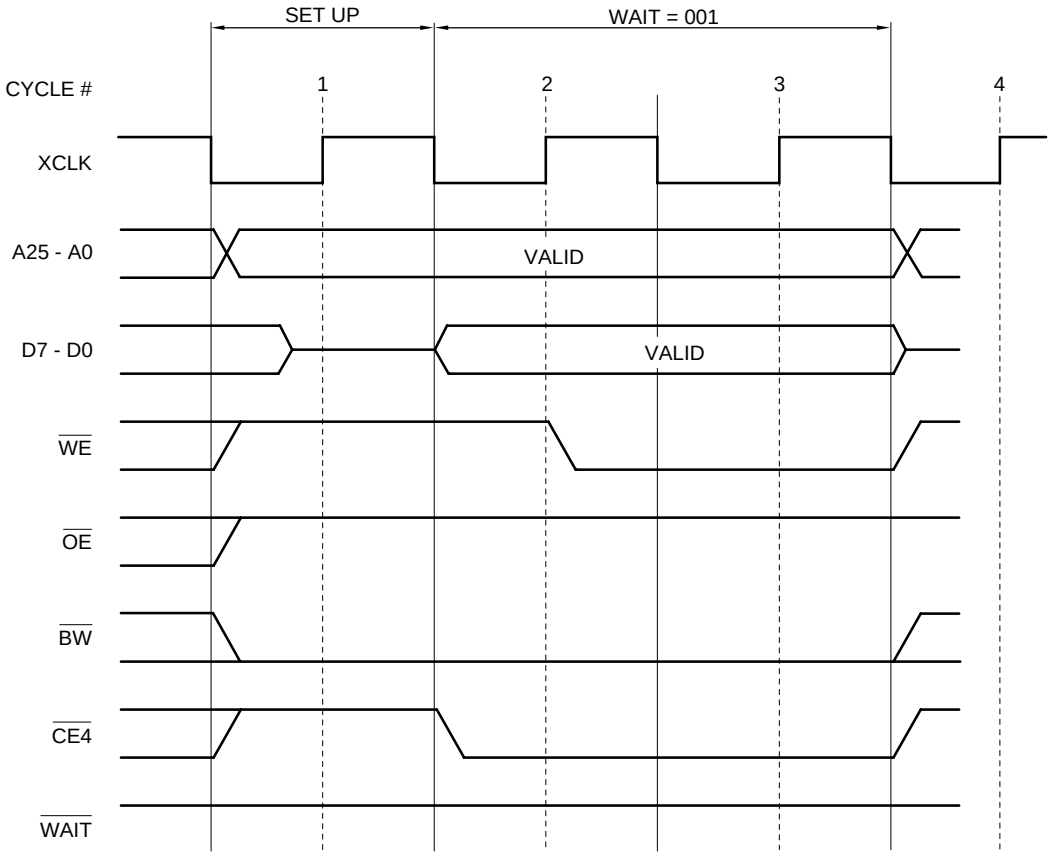


Figure 5-5. Byte Read from x8 SRAM



NOTE: $\overline{CE4}$ selects x8 SRAM

ARM2-89

Figure 5-6. Byte Write to x8 SRAM

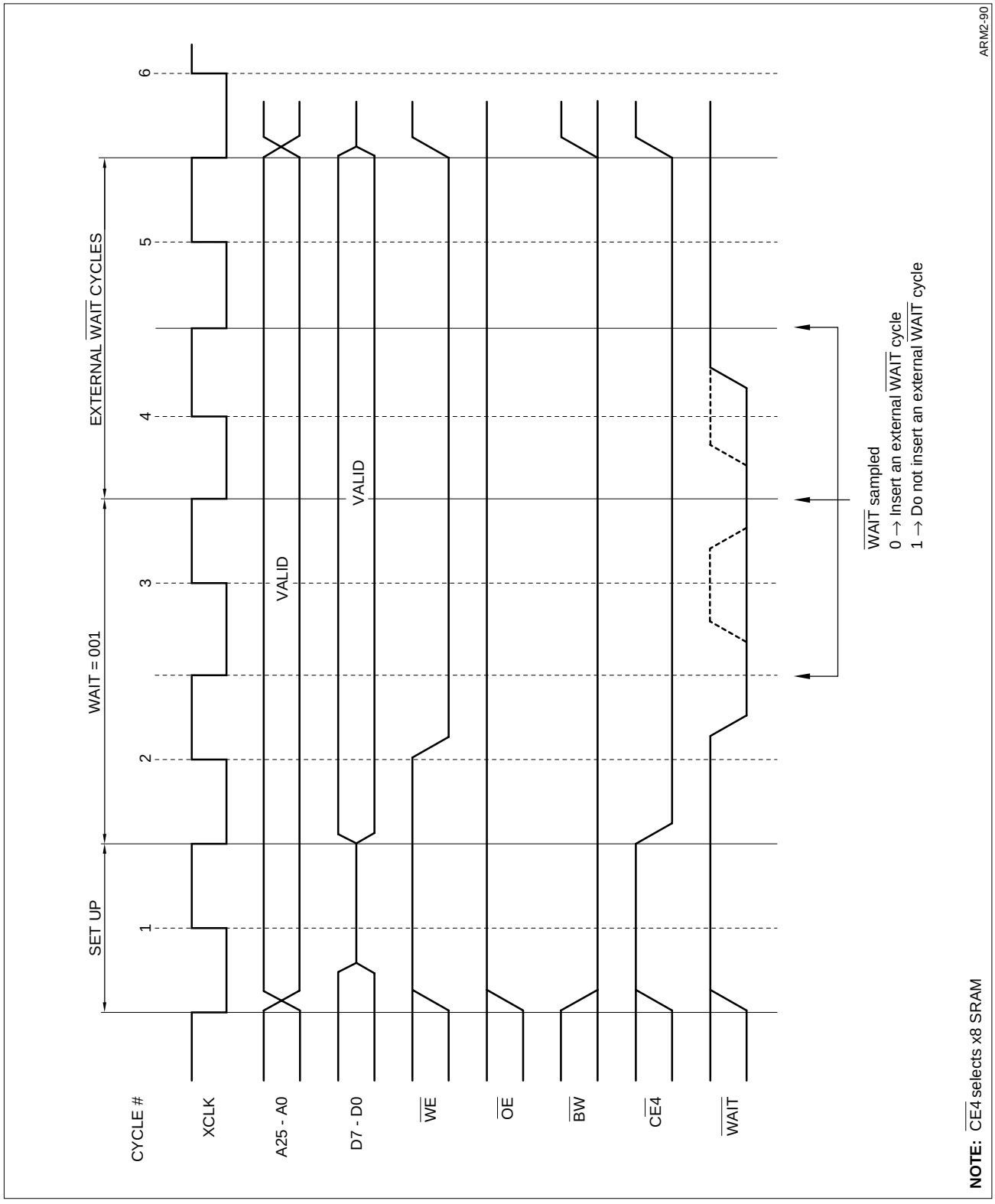
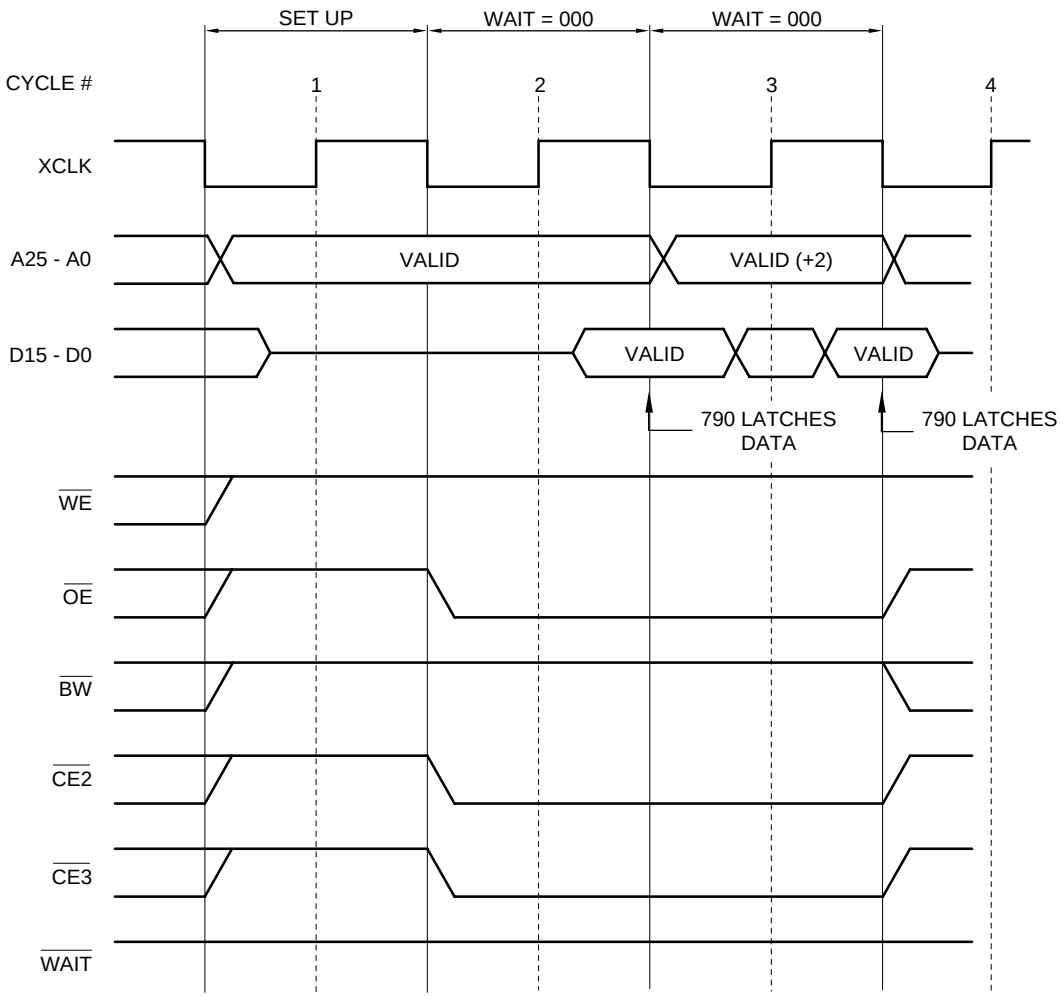


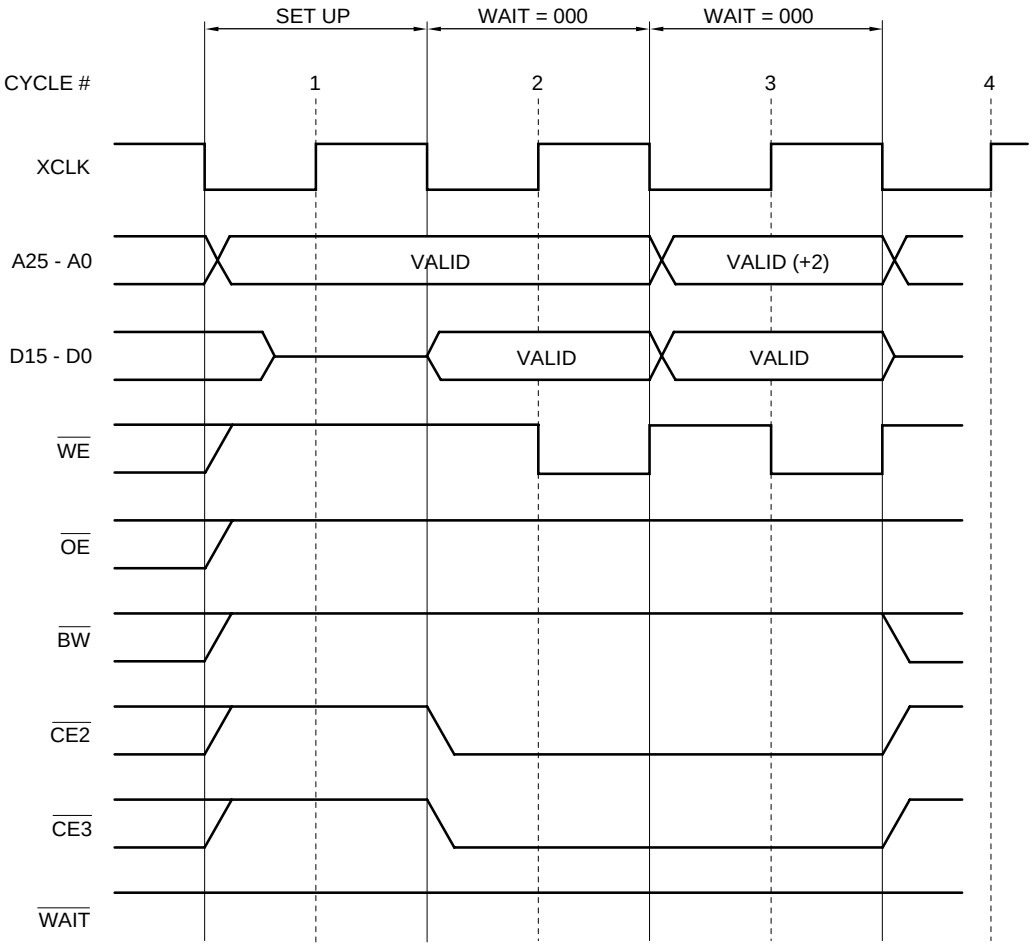
Figure 5-7. Byte Write to x8 SRAM with External Wait Cycles



NOTES:
1. If Half Word mode is enabled (HW = 1), only cycles 1 and 2 will take place. The READ operation will terminate by the end of cycle 2.
2. $\overline{CE2}$ selects LOW byte
3. $\overline{CE3}$ selects HIGH byte

ARM2-91

Figure 5-8. Word Read to x16 SRAM



NOTES:
1. If Half Word Mode is enabled (HW = 1), only cycles 1 and 2 will take place. The WRITE operation will terminate by the end of cycle 2.
2. CE2 selects LOW byte
3. CE3 selects HIGH byte

ARM2-92

Figure 5-9. Word Write to x16 SRAM

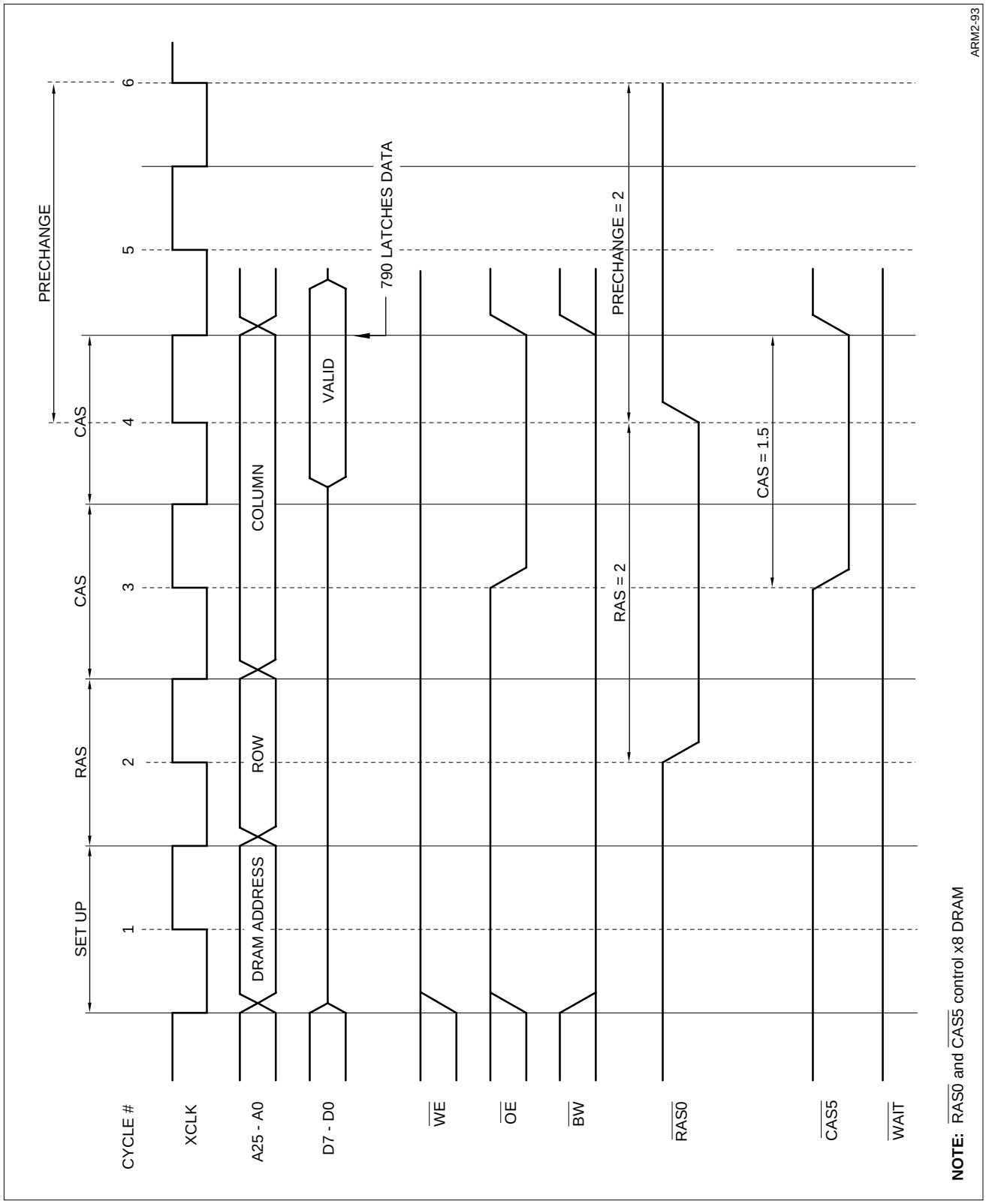


Figure 5-10. Byte Read from x8 DRAM

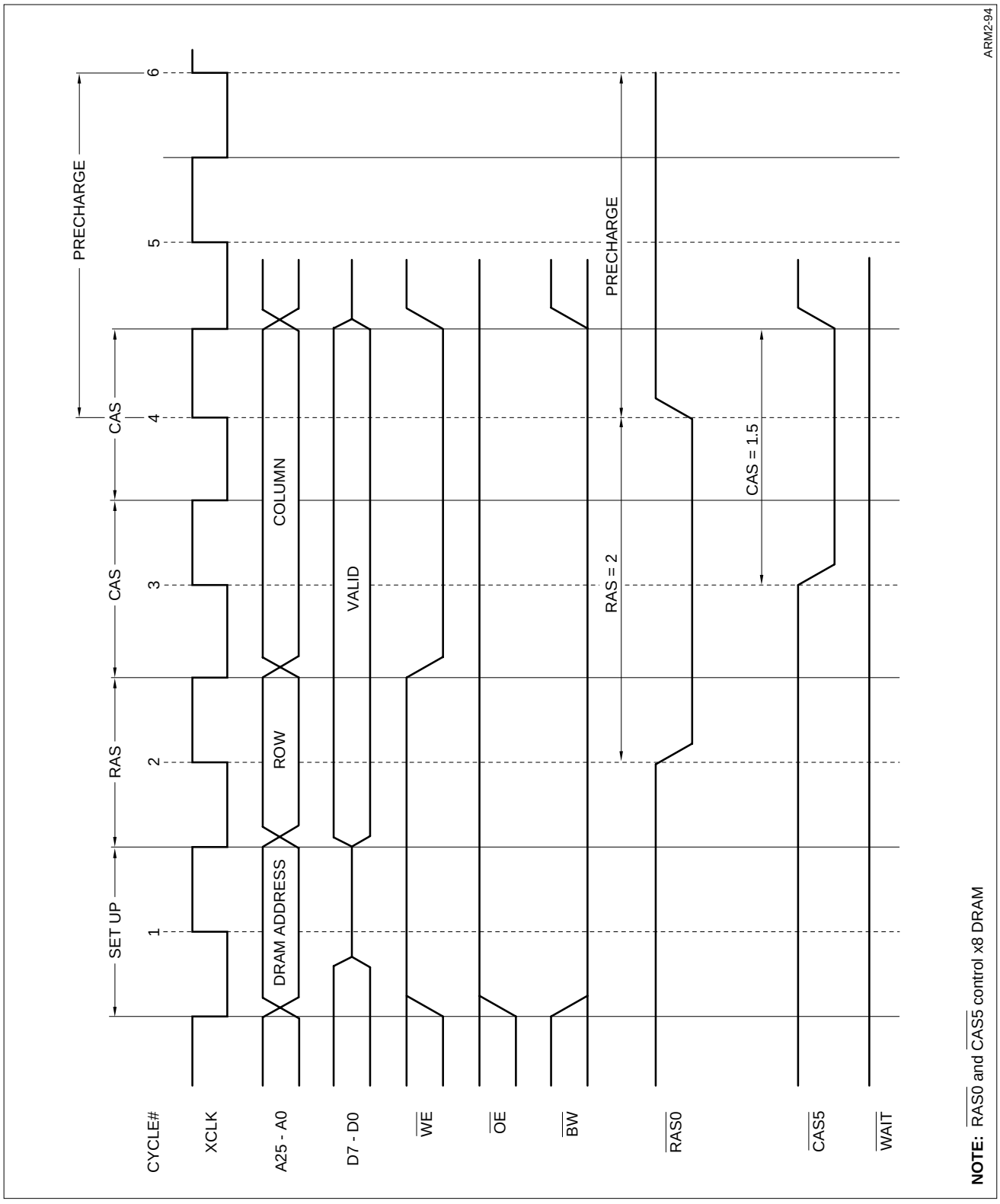


Figure 5-11. Byte Write to x8 DRAM

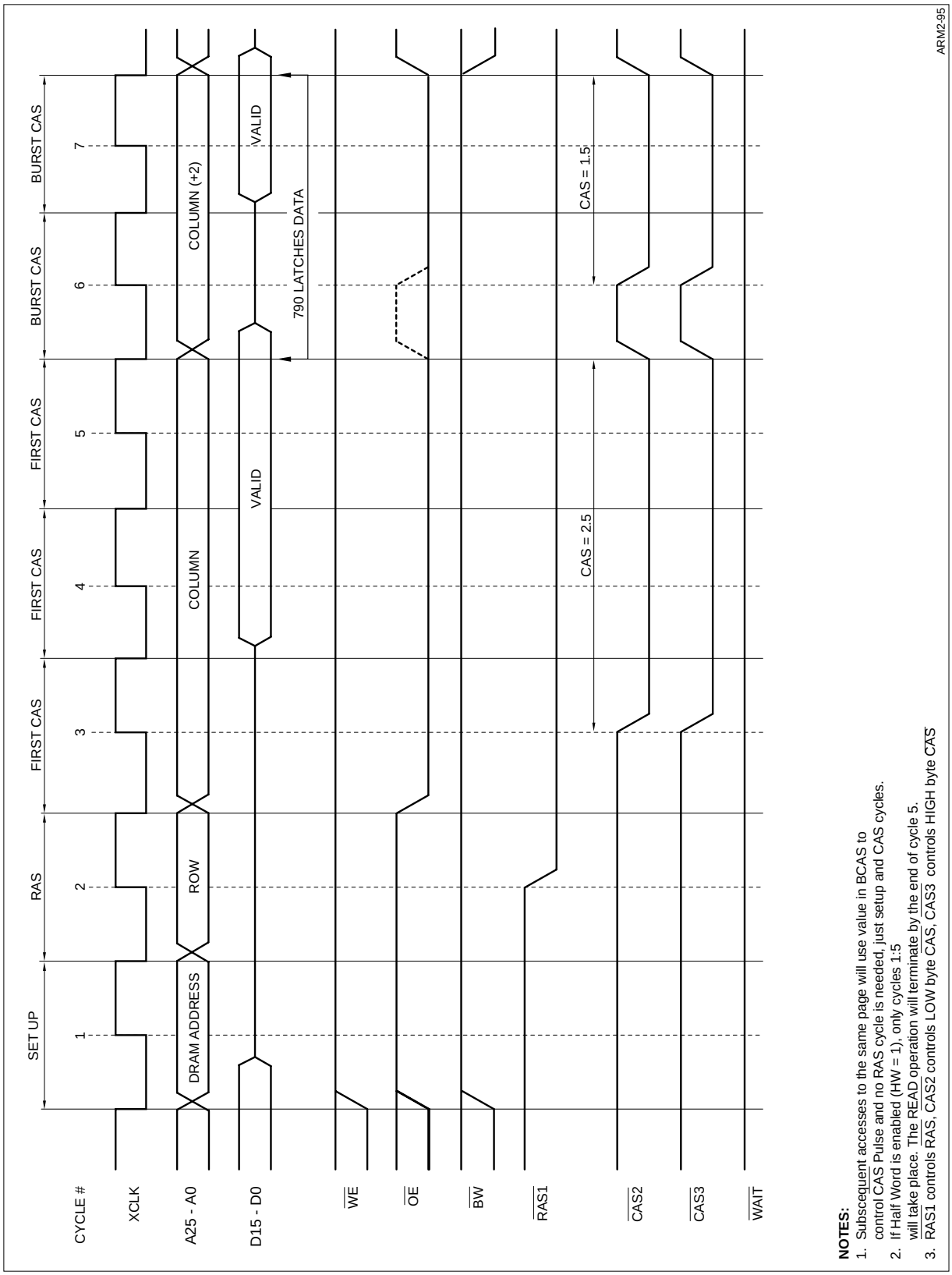


Figure 5-12. Word Read from x16 DRAM Page Mode (FCAS = 010, BCAS = 01)

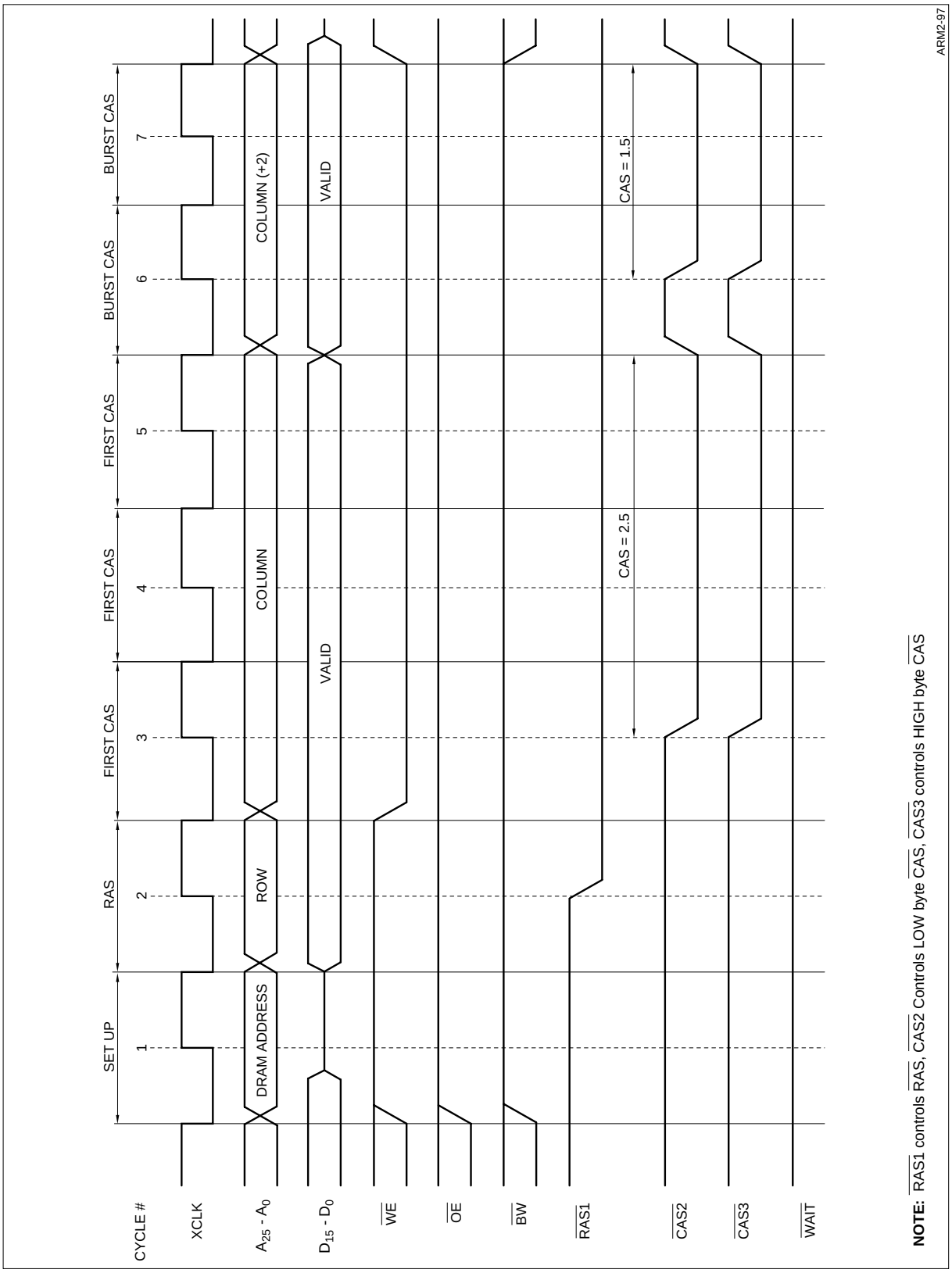


Figure 5-13. Word Write to x16 DRAM Page Mode (FCAS = 010, BCAS = 01)

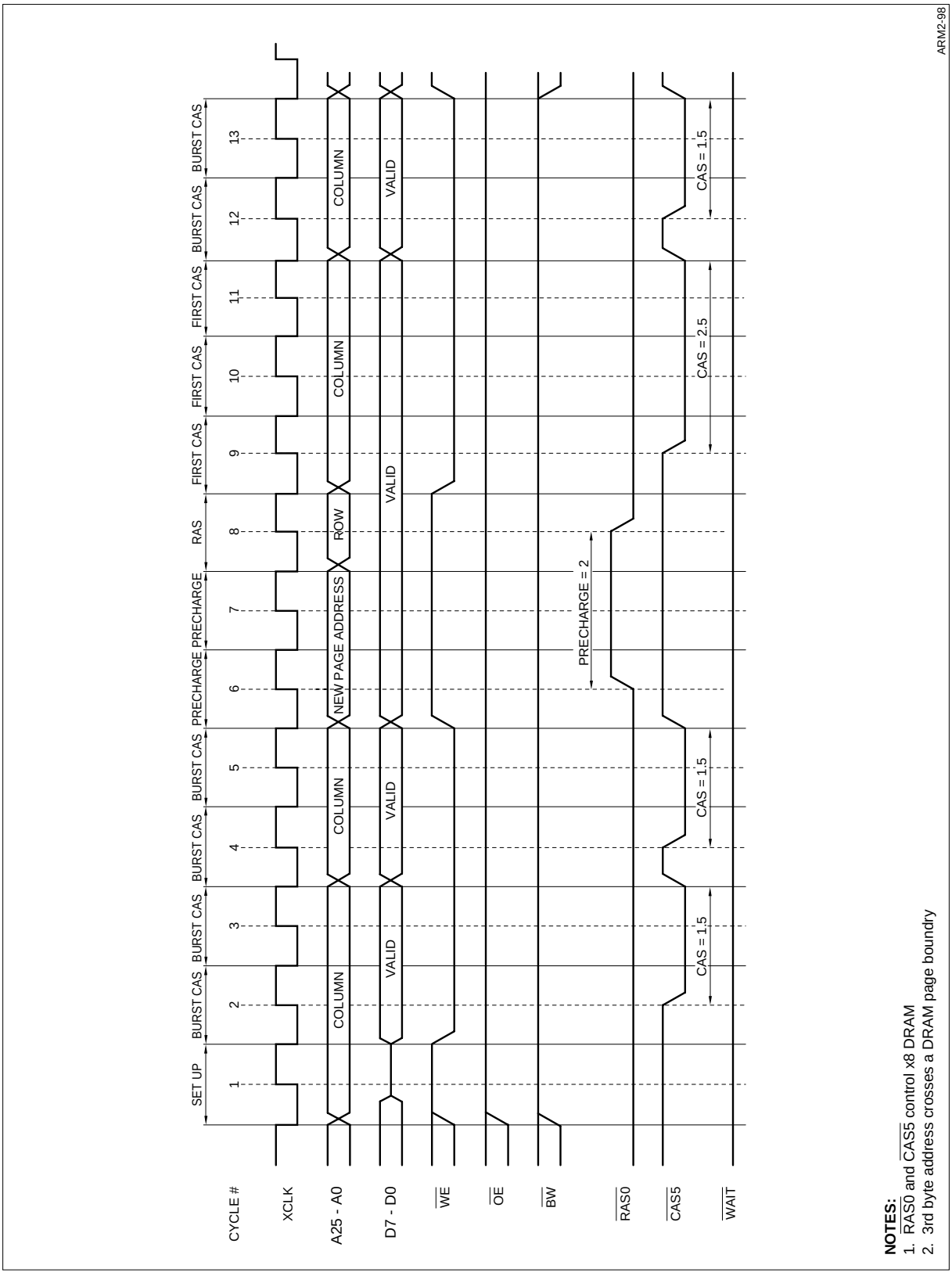


Figure 5-14. Word Write to x8 DRAM Page Mode (FCAS = 010, BCAS = 01) DRAM Page Boundary Crossing

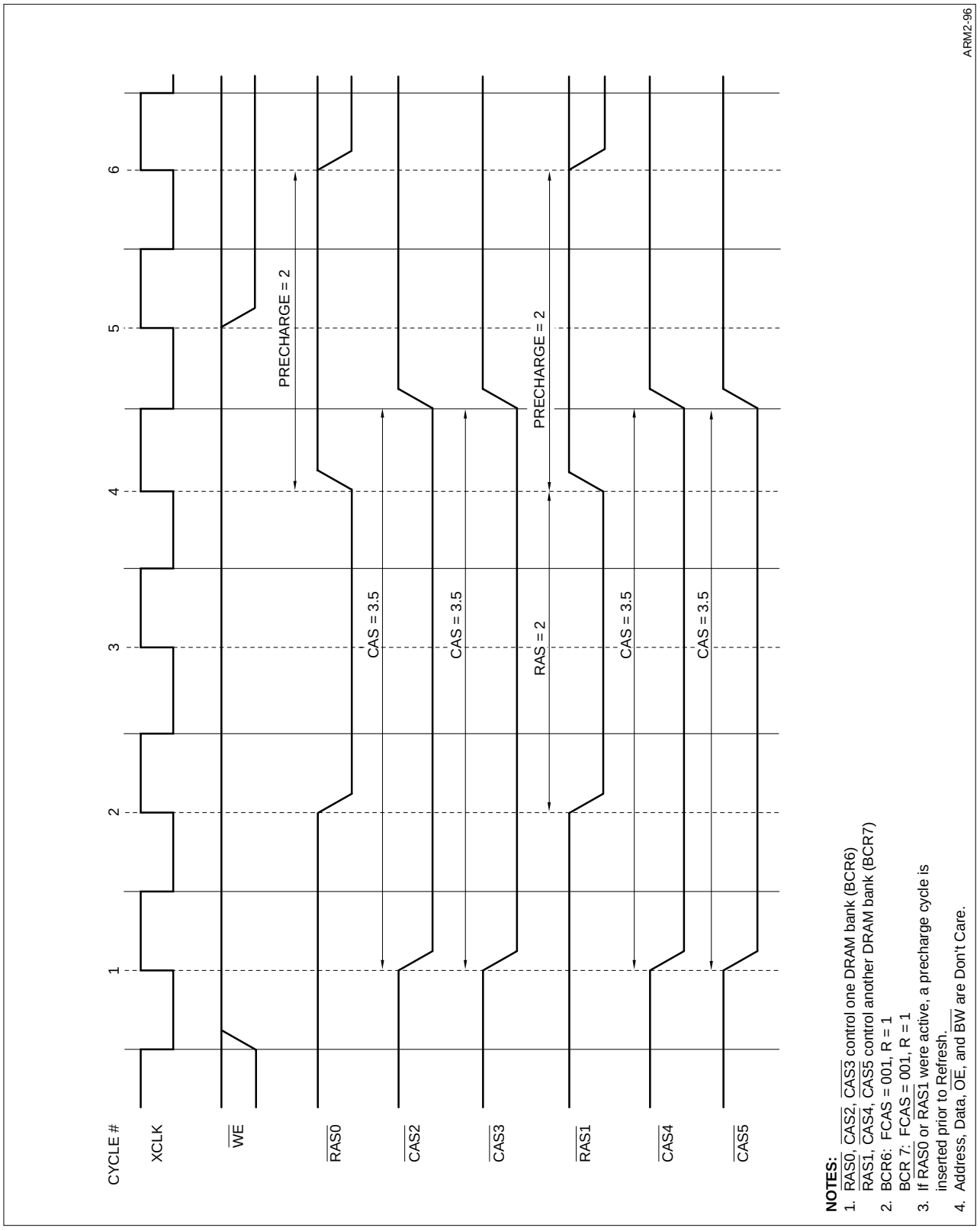


Figure 5-15. DRAM $\overline{\text{CAS}}$ before $\overline{\text{RAS}}$ Refresh Cycle

Chapter 6

CLOCK AND POWER MANAGEMENT

INTRODUCTION

The Clock and Power Management Unit (CPMU) manages the clock and power to the ARM7DI and peripherals. It provides a programmable clock divider to the ARM7DI and peripherals that serve to reduce power. Also, the ARM7DI and peripheral can have their clocks stopped to further reduce power. The ARM7DI clock, when halted, will re-start at the fastest speed when an interrupt to the ARM7DI is detected (either IRQ or FIQ). Power can be further reduced by stopping the external clock via XCLKDIS output.

FEATURES

- Clock Control and Power Management
- Provide Clocks to the ARM7DI and Peripherals
- Provide a Programmable Clock Divider to the ARM7DI
- Provide a Programmable Clock Divider to UARTs and Counters/Timers (LCD Controller and PWMs have Built-In Programmable Clock Dividers)
- Allows External Clocks to UARTs and Counters/Timers
- Stop Clocks to the ARM7DI and/or Peripherals
- Stop External Clock via XCLKDIS
- ARM7DI Clock is Re-started when an Interrupt is Detected
- Power Modes: Active, Standby, Sleep and Stop

BLOCK DIAGRAM

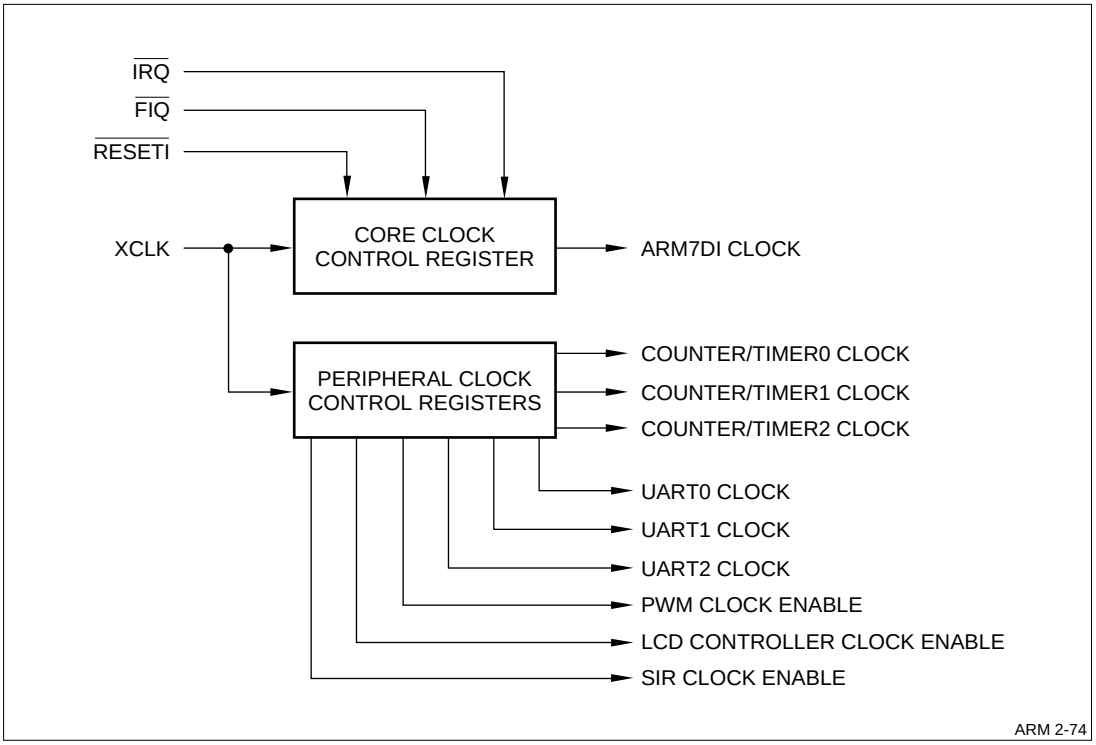


Figure 6-1. Clock and Power Management Unit Block Diagram

REGISTER MAP

The base address for CPMU registers is 0xFFFFAC00. The offset, initial value, access and number of bits for each register are as follows.

Table 6-1.
CPMU Register Offset

CPMU REGISTER	ADDRESS OFFSET [7:0]	RESET VALUE	ACCESS	NUMBER OF BITS
PCSR	04	0x00	R/W	9
U0CCR	08	0x01	R/W	9
U1CCR	0C	0x01	R/W	9
U2CCR	10	0x01	R/W	9
CT0CCR	18	0x01	R/W	9
CT1CCR	1C	0x01	R/W	9
CT2CCR	20	0x01	R/W	9
CCCR	28	0x01	R/W	8

POWER MODES

The 790A contains power management logic to reduce power consumption. Table 6-2 summarizes the four power mode

Table 6-2.
LH77790A Power Modes

FUNCTION	ACTIVE MODE	STANDBY MODE	SLEEP MODE	STOP MODE[1]
External Clock	ON	ON	ON	Halted
ARM7DI & On-Chip Peripherals Clocks	Programmable	Halted	Halted	Halted
DRAM Refresh	ON/OFF	ON	OFF	OFF
Watchdog Timer[2] (WDT)	ON/OFF	OFF	OFF	OFF
XCLKDIS Output (Active HIGH)	OFF (LOW)	OFF (LOW)	ON (HIGH)	ON (HIGH)
Exit Conditions	–	External Interrupt	External Interrupt	External Interrupt
Power Consumption	Normal	→	→	Ultra Low

NOTES:

1. XCLKDIS signal must be controlling the external (Off-Chip) clock.
2. WDT must be OFF to go into the various power-down modes

Active Mode

This mode is the operating mode. The external clock, XCLK, is active. The ARM7DI and On-Chip Peripherals clocks are programmable via the Clock and Power Mana-gement Unit (CPMU) registers and the registers in each Peripheral. DRAM Refresh and WDT can be enabled or disabled. The clock disable output, XCLKDIS, is LOW. This mode consumes normal power. After reset, the 790A is in Active mode.

Standby Mode:

The 790A goes into Standby mode when:

1. ARM7DI clock is halted (By programming CPMU Register CCCR = 0x00), and
2. DRAM Refresh is enabled, and
3. External clock is still active, and
4. WDT is Disabled

Halting the ARM7DI clock will automatically halt all the clocks to the On-Chip Peripherals regardless of what is programmed in the other CPMU and Peripheral registers. Only logic needed for DRAM Refresh will be active. The clock disable output, XCLKDIS, is LOW.

An external interrupt will restart the ARM7DI clock at the external clock (XCLK) speed and the 790A will operate in Active mode.

Sleep Mode

The 790A goes into Sleep mode when:

1. ARM7DI clock is halted (By programming CPMU Register CCCR = 0x00), and
2. DRAM Refresh is disabled, and
3. External clock is still active (XCLKDIS, is not physically connected to the external clock circuitry), and
4. WDT is Disabled

Halting the ARM7DI clock will automatically halt all the clocks to the On-Chip Peripherals regardless of what is programmed in the other CPMU and Peripheral registers. The clock disable output, XCLKDIS, is HIGH but is not physically connected to the external clock circuitry.

An external interrupt will disable XCLKDIS (becomes LOW), allows the external clock circuitry to stabilize (at least 32 clocks) and then restart the ARM7DI clock at the external clock (XCLK) speed. The 790A will operate in Active mode.

Stop Mode

1. ARM7DI clock is halted (By programming CPMU Register CCCR = 0x00), and
2. DRAM Refresh is disabled, and
3. External clock is halted (XCLKDIS, is physically connected to the external clock circuitry), and
4. WDT is Disabled

Halting the ARM7DI clock will automatically halt all the clocks to the On-Chip Peripherals regardless of what is programmed in the other CPMU and Peripheral registers. The clock disable output, XCLKDIS, is HIGH and is physically connected to the external clock circuitry. This mode consumes the lowest power (micro-amps range).

An external interrupt will disable XCLKDIS (becomes LOW), allows the external clock circuitry to stabilize (at least 32 clocks) and then restart the ARM7DI clock at the external clock (XCLK) speed and the 790A will operate in Active mode.

XCLKDIS Output

XCLKDIS is an active high output pin that can be used to disable external clock circuitry and will result in reducing current consumption to micro-amperes. XCLKDIS is HIGH in Sleep and Stop modes. Connecting this pin to the external clock circuitry (see Fig 6-2) allows the 790A to go into Stop mode by disabling the external clock.

NOTE: Prior to going into Standby, Sleep or Stop modes, it is the user's responsibility to have all external I/O in a known state, save the content of DRAM, follow LCD On/Off power sequence and perform any other house-keeping steps.

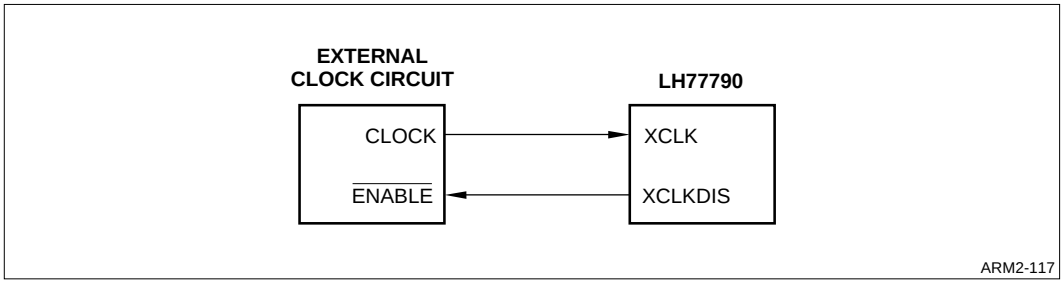


Figure 6-2. External Clock Circuitry Example

CPMU REGISTERS

Peripheral Clock Select Register (PCSR)

8	7	6	5	4	3	2	1	0
SIRCE	PWMCE	LCDCE	CT2CS	CT1CS	CT0CS	UC2S	UC1S	UC0S

The PCSR register has nine bits to control the source of the UARTs and Counters/Timers clocks and enables or disables the clocks for the LCD controller, PWMs, and IRs as follows:

Table 6-3.
PCSR Fields

BIT POSITION	NAME	DESCRIPTION
0	U0CS	UART0 Clock Source: 0 ---> Clock provided internally by the CPMU. 1 ---> Clock provided externally by UCLK input pin. U0CCR has no effect on the External Clock.
1	U1CS	UART1 Clock Source: 0 ---> Clock provided internally by the CPMU. 1 ---> Clock provided externally by UCLK input pin. U1CCR has no effect on the External Clock.
2	U2CS	UART2 Clock Source: 0 ---> Clock provided internally by the CPMU. 1 ---> Clock provided externally by UCLK input pin. U2CCR has no effect on the External Clock.
3	CT0CS	Counter/Timer0 Clock Source: 0 ---> Clock provided internally by the CPMU. 1 ---> Clock provided externally by CTCLK input pin. CT0CCR has no effect on the External Clock.
4	CT1CS	Counter/Timer1 Clock Source: 0 ---> Clock provided internally by the CPMU. 1 ---> Clock provided externally by CTCLK input pin. CT1CCR has no effect on the External Clock.
5	CT2CS	Counter/Timer2 Clock Source: 0 ---> Clock provided internally by the CPMU. 1 ---> Clock provided externally by CTCLK input pin. CT2CCR has no effect on the External Clock.
6	LCDCE*	LCD Controller Clock Enable: 0 ---> Clock Disabled. 1 ---> Clock Enabled and is equal to XCLK. LCD Controller generates its own divided clock.
7	PWMCE	PWM Clock Enable: 0 ---> Clock Disabled. 1 ---> Clock Enabled and is equal to XCLK. This bit controls the clocks for all three PWMs. All PWMs generate their own divided clocks from XCLK.
8	SIRCE	Serial InfraRed Clock Enable: 0 ---> Clock Disabled. 1 ---> Clock Enabled. The enable bit is sent to the SIR unit to enable/disable its own clocks as defined in the SIR chapter.

NOTE: *Stopping the LCD Controller clock will cause the LCD panel control signals to stop, so care must be taken to insure that the power on/off sequence is observed to prevent DC build-up and panel damage.

UART0 Clock Control Register (U0CCR)

This register controls UART0 clock frequency shown in Table 6-4.

Table 6-4.
UART0 Clock Frequency

U0CCR	CLOCK DIVISOR	UART0 CLOCK FREQUENCY
000000000	0	Halted
000000001	1	XCLK
000000010	2	XCLK/2
000000100	4	XCLK/4
000001000	8	XCLK/8
000010000	16	XCLK/16
000100000	32	XCLK/32
001000000	64	XCLK/64
010000000	128	XCLK/128
100000000	256	XCLK/256

The bits in the above register are mutually exclusive meaning that only one bit should be programmed to a value '1' for valid operation. When a valid value is programmed into this register, the generated clock (with the new frequency) will be glitch-free. If more than one bit is programmed to a value '1', the specification is violated and *undefined operation* will occur.

UART1 Clock Control Register (U1CCR)

This register controls UART1 clock frequency as shown in Table 6-5.

Table 6-5.
UART1 Clock Frequency

U1CCR	CLOCK DIVISOR	UART1 CLOCK FREQUENCY
000000000	0	Halted
000000001	1	XCLK
000000010	2	XCLK/2
000000100	4	XCLK/4
000001000	8	XCLK/8
000010000	16	XCLK/16
000100000	32	XCLK/32
001000000	64	XCLK/64
010000000	128	XCLK/128
100000000	256	XCLK/256

The bits in the above register are mutually exclusive meaning that only one bit should be programmed to a value '1' for valid operation. When a valid value is programmed into this register, the generated clock (with the new frequency) will be glitch free. If more than one bit is programmed to a value '1', the specification is violated and *undefined operation* will occur.

UART2 Clock Control Register (U2CCR)

This register controls UART2 clock frequency as shown in Table 6-6.

Table 6-6.
UART2 Clock Frequency

U2CCR	CLOCK DIVISOR VALUE (D[7:0])	UART2 CLOCK FREQUENCY
000000000	0	Halted
000000001	1	XCLK
000000010	2	XCLK/2
000000100	4	XCLK/4
000001000	8	XCLK/8
000010000	16	XCLK/16
000100000	32	XCLK/32
001000000	64	XCLK/64
010000000	128	XCLK/128
100000000	256	XCLK/256

The bits in the above register are mutually exclusive meaning that only one bit should be programmed to a value '1' for valid operation. When a valid value is programmed into

this register, the generated clock (with the new frequency) will be glitch-free. If more than one bit is programmed to a value '1', the specification is violated and *undefined operation* will occur.

Counter/Timer0 Clock Control Register (CT0CCR)

This register controls Counter/Timer0 clock frequency as shown in Table 6-7.

Table 6-7.
Counter/Timer0 Clock Frequency

CT0CCR	CLOCK DIVISOR	COUNTER/TIMER0 CLOCK FREQUENCY
000000000	0	Halted
000000001	1	XCLK
000000010	2	XCLK/2
000000100	4	XCLK/4
000001000	8	XCLK/8
000010000	16	XCLK/16
000100000	32	XCLK/32
001000000	64	XCLK/64
010000000	128	XCLK/128
100000000	256	XCLK/256

The bits in the above register are mutually exclusive meaning that only one bit should be programmed to a value '1' for valid operation. When a valid value is programmed into this register, the generated clock (with the new frequency) will be glitch-free. If more than one bit is programmed to a value '1', the specification is violated and *undefined operation* will occur.

Counter/Timer1 Clock Control Register (CT1CCR)

This register controls Counter/Timer1 clock frequency as shown in Table 6-8.

Table 6-8.
Counter/Timer1 Clock Frequency

CT1CCR	CLOCK DIVISOR	COUNTER/TIMER1 CLOCK FREQUENCY
000000000	0	Halted
000000001	1	XCLK
000000010	2	XCLK/2
000000100	4	XCLK/4
000001000	8	XCLK/8
000010000	16	XCLK/16
000100000	32	XCLK/32
001000000	64	XCLK/64
010000000	128	XCLK/128
100000000	256	XCLK/256

The bits in the above register are mutually exclusive meaning that only one bit should be programmed to a value '1' for valid operation. When a valid value is programmed into this register, the generated clock (with the new frequency) will be glitch-free. If more than one bit is programmed to a value '1', the specification is violated and *undefined operation* will occur.

Counter/Timer2 Clock Control Register (CT2CCR)

This register controls Counter/Timer2 clock frequency as shown in Table 6-9.

Table 6-9.
CounterTimer2 Clock Frequency

CT2CCR	CLOCK DIVISOR	COUNTER/TIMER2 CLOCK FREQUENCY
000000000	0	Halted
000000001	1	XCLK
000000010	2	XCLK/2
000000100	4	XCLK/4
000001000	8	XCLK/8
000010000	16	XCLK/16
000100000	32	XCLK/32
001000000	64	XCLK/64
010000000	128	XCLK/128
100000000	256	XCLK/256

The bits in the above register are mutually exclusive meaning that only one bit should be programmed to a value '1' for valid operation. When a valid value is programmed into this register, the generated clock (with the new frequency) will be glitch-free. If more

than one bit is programmed to a value '1', the specification is violated and *undefined operation* will occur.

Core Clock Control Register (CCCR)

This register controls the ARM7DI clock frequency as shown in Table 6-10.

Table 6-10.
CPU Clock Frequency

CCCR	CLOCK DIVISOR	ARM7DI CLOCK FREQUENCY
00000000	0	Halted*
00000001	1	XCLK
00000010	2	XCLK/2
00000100	4	XCLK/4
00001000	8	XCLK/8
00010000	16	XCLK/16
00100000	32	XCLK/32
01000000	64	XCLK/64
10000000	128	XCLK/128

NOTE: *Halting the ARM7DI clock will disable clocks to all functional units (UARTs, Counter/Timers, ...) including the LCD controller. Care must be taken to insure that proper power on/off is observed to prevent DC build-up and LCD panel damage.

The ARM7DI clock, when halted, will re-start at the fastest speed (XCLK) when an interrupt to the ARM7DI is detected (either IRQ or FIQ).

The bits in the above register are mutually exclusive, meaning that only one bit should be programmed to a value '1' for valid operation. When a valid value is programmed into this register, the generated clock (with the new frequency) is glitch-free. If more than one bit is programmed to a value '1', the specification is violated and an *undefined operation* occurs.

Chapter 7

UART

INTRODUCTION

The 790A provides three 16C450 class UART functional blocks (Universal Asynchronous Receiver/Transmitter). UART0 supports a nine-wire interface (TxD, RxD, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, $\overline{\text{DTR}}$, $\overline{\text{DSR}}$, $\overline{\text{DCD}}$, $\overline{\text{RI}}$, and GND). UART1 supports a six-wire interface (TxD, RxD, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, $\overline{\text{RI}}$, and GND). UART2 supports a three-wire interface (TxD, RxD, and GND) and is IrDA 1.0/DASK IR compliant.

FEATURES

- Double Buffering
- 16-bit Programmable Baud Generator
- Baud Generator drives the Transmitter and Receiver Clocks
- Internal as well as External Input Clocks
- Independently Controlled Interrupts
  - Receive, Transmit, Line Status, and Modem Status
- Priority Interrupts
- Programmable Data Length, Stop Bits and Parity
- Complete Status Reporting
- False Start Bit Detection
- Line Break Generation and Detection
- Full Modem Support on UART0
- Loop Back Mode
- IrDA IR support on Channel 2
- SHARP DASK IR (Digital Amplitude Shift Keying) Support on Channel 2

BLOCK DIAGRAM

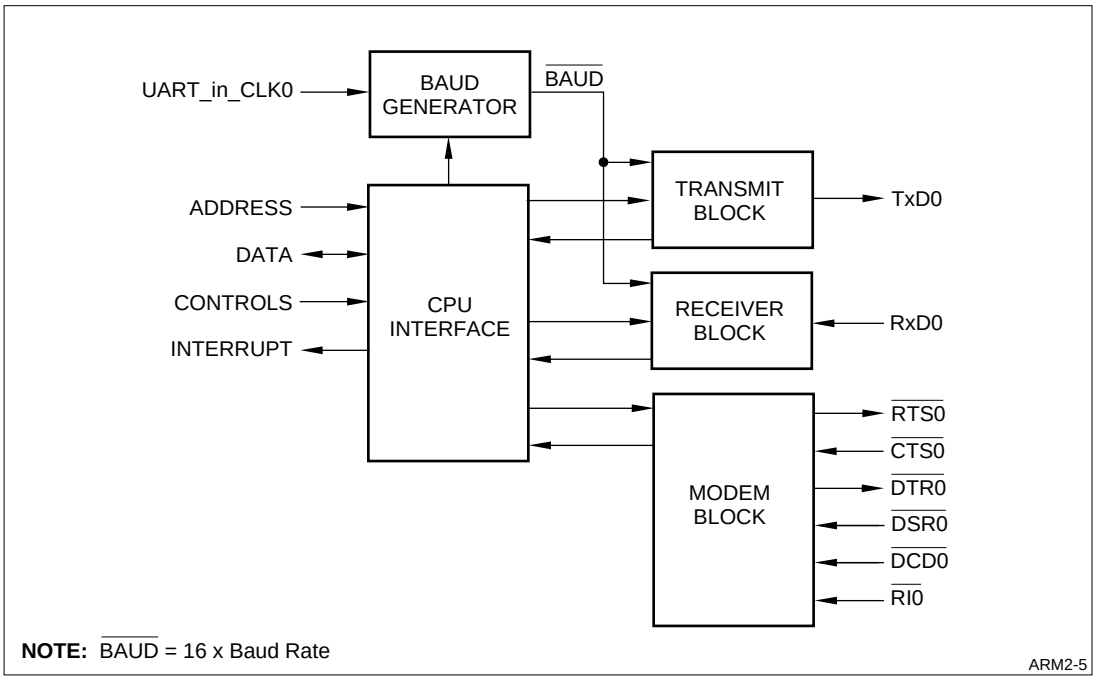


Figure 7-1. UART0 Block Diagram

NOTE: UART1 and UART2 have similar block diagrams except that UART1 does not support $\overline{DTR}$, $\overline{DSR}$, $\overline{DCD}$, and UART2 only supports TxD and RxD.

UPON RESET

UART2 will act as an IrDA port coming out of reset.

GENERAL OPERATION

The UART consists of four functional blocks: Baud Generator, Transmitter, Receiver, and Modem Controller.

Baud Generator

The UART has a programmable baud generator that is capable of dividing the input clock by a divisor from 1 to 2^{16} and producing a clock frequency, $\overline{BAUD}$, that is 16 times faster than the desired baud rate for driving both the transmitter and the receiver blocks. The $\overline{BAUD}$ is then divided by 16 internally to the receiver and the transmitter to generate the required baud rate clock frequency.

Transmitter

The transmitter section consists of the Transmitter Holding Register (THR) and the Transmitter Shift Register (TSR). Transmission is initiated by writing the data to be sent to the Transmitter Holding Register (THR). The data is then transferred to the Transmitter Shift Register (TSR) when it is empty, together with a START bit, PARITY bit, and STOP bit, as determined by the Line Control Register (LCR). The bits to be transmitted are then clocked out of the TSR starting with bit (0), via TxD, by the $\overline{BAUD}$ clock.

If enabled, an interrupt is generated when the Transmitter Holding Register becomes empty.

Receiver

The Receiver section consists of the Receiver Shift Register (RSR) and the Receiver Buffer Register (RBR). Data received on the RxD input are clocked into the Receiver Shift Register (RSR) by the $\overline{BAUD}$ clock. First bit received is bit (0) A filter is used to remove spurious inputs which last for less than two periods of $\overline{BAUD}$. When the complete character has been clocked into the RSR, the data bits are then transferred to the Receiver Buffer Register (RBR) to be read by the ARM7DI. The receiver also checks for a STOP bit and for correct PARITY as determined by the Line Control Register (LCR).

If enabled, an interrupt will be generated when the data have been transferred to the RBR. Interrupts can also be generated for incorrect PARITY, a missing STOP bit, an overrun error, or a line break if enabled.

Due to the fact that the receiver is double buffered (RSR and RBR), the UART can receive another character into the RSR while the character in the RBR is being read by the ARM7DI. Note that software has no access to the RSR.

Modem Controller

The output modem control lines ($\overline{RTS}$ and $\overline{DTR}$) can be set or cleared by writing to the Modem Control Register (MCR). The current status of the input modem status lines ($\overline{DCD}$, $\overline{RI}$, $\overline{DSR}$, $\overline{CTS}$) can be read from the Modem Status Register (MSR).

If enabled, an interrupt will be generated when $\overline{DSR}$, $\overline{CTS}$, or $\overline{DCD}$ change states or $\overline{RI}$ making a transition from LOW to HIGH.

Modem Signals

All signals other than RxD, and TxD will be shared (multiplexed) with the PPI signals. The multiplexing information and the required configuration of the PPI signals when it is used as modem signals are detailed in the I/O Configuration chapter. When any of the multiplexed signals is not selected to function as a modem signal, it is undefined as far as the UART is concerned.

Modem signals not supported by UART1 and UART2 are undefined.

EXTERNAL INTERFACE

Table 7-1.
UART External Interface

SIGNAL	BIT	INPUT/OUTPUT	FUNCTION
TxD0, TxD1, TxD2	3	Output	Serial Data Output (Transmitted Data). Upon Reset, TxD = 1.
RxD0, RxD1, RxD2	3	Input	Serial Data Input (Received Data).
$\overline{RTS0}$, $\overline{RTS1}$	2	Output	Request to Send. Active LOW. Upon Reset, $\overline{RTS}$ = 1. Supported on UART0 and UART1.
$\overline{CTS0}$, $\overline{CTS1}$	2	Input	Clear to Send. Active LOW. Supported on UART0 and UART1.
$\overline{DTR0}$	1	Output	Data Terminal Ready. Active LOW. Upon Reset $\overline{DTR0}$=1. Supported on UART0 only.
$\overline{DSR0}$	1	Input	Data Set Ready. Active LOW. Supported on UART0 only.
$\overline{RI0}$, $\overline{RI1}$	2	Input	Ring Indicator. Active LOW. Supported on UART0 and UART1.
$\overline{DCD0}$	1	Input	Data Carrier Detect. Active LOW. Supported on UART0 only.

REGISTER MAP

The base addresses for UART0 is 0xFFFF0000. The offset, initial value, access and number of bits for each register is as follows:

Table 7-2.
UART Register Map

UART0 REGISTER	ADDRESS OFFSET [7:0]	RESET VALUE	ACCESS	NUMBER OF BITS
RBR0 THR0 DLL0	0x00 0x00 0x00	Undefined Undefined Undefined	Read Write R/W	8 8 8
IER0 DLM0	0x04 0x04	0x00 Undefined	R/W R/W	8 8
IIR0	0x08	0x01	Read	8
LCR0	0x0C	0x00	R/W	8
MCR0	0x10	0x00	R/W	8
LSR0	0x14	0x60	Read	8
MSR0	0x18	MSR[3:0] = 0 MSR[7:4] = Undefined	R/W	8
SCR0	0x1C	Undefined	R/W	8

The base addresses for UART1 register are 0xFFFF0400. The offset, initial value, access and number of bits for each register is as follows:

Table 7-3.
UART Register Map

UART1 REGISTER	ADDRESS OFFSET [7:0]	RESET VALUE	ACCESS	NUMBER OF BITS
RBR1 THR1 DLL1	0x00 0x00 0x00	Undefined Undefined Undefined	Read Write R/W	8 8 8
IER1 DLM1	0x04 0x04	0x00 Undefined	R/W R/W	8 8
IIR1	0x08	0x01	Read	8
LCR1	0x0C	0x00	R/W	8
MCR1	0x10	0x00	R/W	8
LSR1	0x14	0x60	Read	8
MSR1	0x18	MSR[3:0] = 0 MSR[7:4] = Undefined	R/W	8
SCR1	0x1C	Undefined	R/W	8

The base addresses for UART2 register are 0xFFFF0800. The offset, initial value, access and number of bits for each register is as follows:

Table 7-4.
UART Register Map

UART2 REGISTER	ADDRESS OFFSET [7:0]	RESET VALUE	ACCESS	NUMBER OF BITS
RBR2 THR2 DLL2	0x00 0x00 0x00	Undefined Undefined Undefined	Read Write R/W	8 8 8
IER2 DLM2	0x04 0x04	0x00 Undefined	R/W R/W	8 8
IIR2	0x08	0x01	Read	8
LCR2	0x0C	0x00	R/W	8
MCR2	0x10	0x00	R/W	8
LSR2	0x14	0x60	Read	8
MSR2	0x18	MSR[3:0] = 0 MSR[7:4] = Undefined	R/W	8
SCR2	0x1C	Undefined	R/W	8

UARTs REGISTERS

Table 7-3 summarizes the UART Registers. Note that DLAB (LCR[7]) when set to '1' selects the divisor registers.

Table 7-5.
UART Registers

DLAB LCR(7)	REGISTER NAME	DATA BIT							
		7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	Receiver Buffer (RBR) (Read only)	D7	D6	D5	D4	D3	D2	D1	D0
0	Transmitter Holding (THR) (Write Only)	D7	D6	D5	D4	D3	D2	D1	D0
1	LSB Divisor Latch (DLL)	D7	D6	D5	D4	D3	D2	D1	D0
1	MSB Divisor Latch (DLM)	D15	D14	D13	D12	D11	D10	D9	D8
0	Interrupt Enable (IER)	_	_	_	_	Enable Modem Status Interrupt (EDSSI)	Enable Receiver Status Interrupt (ELSI)	Enable THR Empty Interrupt (ETBEI)	Enable Receiver Buffer Interrupt (ERBFI)
X	Interrupt Identification (IIR) (Read Only)	0	0	0	0	0	Interrupt Identifier 00 = Modem Status 01 = THR Empty 10 = Receiver Buffer 11 = Receiver Line Status		0 = Interrupt Pending 1 = No Interrupt Pending
X	Line Control (LCR)	DLAB	Set Break (SB)	PARITY Type 00 = Odd 01 = Even 10 = Force 1 11 = Force 0		PARITY Enable (PEN)	STOP Bits (STB)	Character Length 00 = 5-bits 01 = 6-bits 10 = 7-bits 11 = 8-bits	
X	Modem Control (MCR)	_	_	_	Loop	OUT2	OUT1	RTS	DTR
X	Line Status (LSR) (Read only)	0	Transmit Empty (TEMT)	THR Empty (THRE)	Break Interrupt (BI)	Frame Error (FE)	PARITY Error (PE)	Overrun Error (OE)	Data Ready (DR)
X	Modem Status (MSR)	Data Carrier Detect (DCD)	Ring Indicator (RI)	Data Set Ready (DSR	Clear To Send (CTS)	Delta DCD (DDCD)	Trailing Edge Ring Indicator (TERI)	Delta DSR (DDSR)	Delta CTS (DCTS)
0	Scratch Pad (SCR)	D7	D6	D5	D4	D3	D2	D1	D0

NOTE: X = Don't Care

REGISTER DESCRIPTION

Receiver Buffer Register (RBR)

The Receiver Buffer Register (RBR) is updated in parallel from the Receiver Shift Register (RSR) at the end of a character receive sequence. If the Receiver Buffer Interrupt (ERBFI) is enabled, an interrupt is generated. This interrupt is cleared in the UART when data is read out of the RBR. RBR is a read only register.

Transmit Holding Register (THR)

The ARM7DI writes the data to be transferred into the Transmit Holding Register (THR). Once the Transmitter Shift Register (TSR) is empty, data in the THR is transferred in parallel to the TSR. If the THR is empty and the THR Empty Interrupt (ETBEI) is enabled, then an interrupt is generated from the UART. The interrupt from the UART is cleared when data is written into THR or the Interrupt Identification Register (IIR) is read. THR is a write only Register.

Interrupt Enable Register (IER)

The Interrupt Enable Register (IER) enables each of the four types of interrupts and the UART interrupt signal to the interrupt controller. The contents of the register are defined in Table 7-6.

Table 7-6.
IER Fields

BIT POSITION	IER CONTROL BIT NAME	DESCRIPTION
0	Enable Receiver Buffer Interrupt (ERBFI)	0 ---> Disable Interrupt. 1 ---> Generate an Interrupt if RBR contains data.
1	Enable Transmit Holding Register Empty Interrupt (ETBEI)	0 ---> Disable Interrupt. 1 ---> Generate an Interrupt if THR is empty or LSR(5)=1.
2	Enable Receiver Line Status Interrupt (ELSI)	0 ---> Disable Interrupt. 1 ---> Generate an Interrupt if LSR(1), LSR(2), LSR(3), or LSR(4) are set.
3	Enable Modem Status Interrupt (EDSSI)	0 ---> Disable Interrupt. 1 ---> Generate an Interrupt if MSR(0), MSR(1), MSR(2), or MSR(3) are set.
4- 7	—	These bits are not used and are set to 0 after Reset.

Interrupt Identification Register (IIR)

The Interrupt Identification Register (IIR) indicates the type of interrupt. The contents of the register are defined in Table 7-7.

Table 7-7.
Interrupt Identification Bit Description

IIR			PRIORIT TYPE	INTERRUP TYPE	CLEAR INTERRUPT
BIT 2	BIT 1	BIT 0			
0	0	1	No Interrupt	No Interrupt	—
1	1	0	1	Receiver Line Status	Reading Line Status Register (LSR)
1	0	0	2	Receiver Buffer	Reading Receiver Buffer Register (RBR)
0	1	0	3	Transmitter Holding Register Empty	Reading Interrupt Identification Register (IIR) OR Writing into the Transmitter Holding Register (THR)
0	0	0	4	Modem Status	Reading Modem Status Register (MSR) OR Writing 0 to MSR[3:0]

NOTE: Bits [3:7]: Not used and are always set to logic 0.

When multiple interrupts are pending, the interrupt line pulses low after each service. IIR is a read only register.

Line Control Register (LCR)

The Line Control Register (LCR) controls the format of data transmitted/ received. The contents of the register are described in Table 7-8.

Table 7-8.
LCR Fields

BIT POSITION	NAME	DESCRIPTION
0, 1	Character Length	The number of bits in transmitted or received serial characters. **Bit 0** **Bit 1** **Character Length** 0 0 5-bits 1 0 6-bits 0 1 7-bits 1 1 8-bits
2	STOP Bits (STB)	The number of STOP bits in each transmitted character. The receiver checks the first STOP bit only, regardless of the number of STOP bits selected. **Bit 2** **STOP Bits** 0 Always one STOP bit 1 1 1/2 STOP bits for 5-bit data, 2 STOP bits for 6, 7, and 8-bit data A STOP bit is a logic 1.
3	PARITY Enable (PEN)	PARITY is generated in transmitted data and is inserted between the last data bit in the character and the first STOP bit. PARITY is checked in received data. **Bit 3** **PARITY** 0 No PARITY generation/check 1 PARITY both generated and checked
5, 4	PARITY Type	Determine PARITY Type when PEN = 1: Odd PARITY: Odd number of 1s are sent checked Even PARITY: Even number of 1s are sent and checked Force 1: PARITY is sent and checked as a logic 1 Force 0: PARITY is sent and checked as a logic 0 **Bit 5** **Bit 4** **PARITY Type** 0 0 Odd 0 1 Even 1 0 Force 1 1 1 Force 0
6	Set Break (SB)	Break Control bit. When set to 1, a break condition is forced. This forces the serial output, TxD, to logic 0. **Bit 6** **TxD State** 0 Normal operation 1 TxD is forced to logic 0
7	Divisor Latch Access Bit (DLAB)	This provides the mechanism to access the two divisor latches for the Baud generator. **Bit 7** **Registers Accessed** 0 Read Receiver Buffer Register (RBR) Write Transmitter Holding Register (THR) Read/Write Interrupt Enable Register (IER) 1 Read/Write Divisor latches (DLL and DLM)

Modem Control Register (MCR)

The Modem Control Register (MCR) controls interfacing to a modem. The contents of the register are described in Table 7-9.

Table 7-9.
MCR Fields

BIT POSITION	NAME	DESCRIPTION
0	DTR	Controls $\overline{DTR}$ Output in normal and loop back operation as follows: 0 = $\overline{DTR}$ output is 1 1 = $\overline{DTR}$ output is 0
1	RTS	Controls $\overline{RTS}$ Output in normal and loop back operation as follows: 0 = $\overline{RTS}$ output is 1 1 = $\overline{RTS}$ output is 0
2	OUT1	Controls $\overline{OUT1}$ Output in normal and loop back operation as follows: 0 = $\overline{OUT1}$ output is 1 1 = $\overline{OUT1}$ output is 0 $\overline{OUT1}$ is not supported as an external output in the 790A
3	OUT2	Controls $\overline{OUT2}$ output in normal and loop back operation as follows: 0 = $\overline{OUT2}$ output is 1 1 = $\overline{OUT2}$ output is 0 $\overline{OUT2}$ is not supported as an external output in the 790A
4	Loop	0 = Normal operation 1 = Loop back mode
5 - 7	—	Not used. Set to 0 after Reset

The following occurs in loop back mode:

1. TxD = 1.
2. RxD is disconnected from the Receiver Shift Register (RSR).
3. The output of the Transmitter Shift Register (TSR) is looped back into the Receiver Shift Register (RSR).
4. The modem status inputs ($\overline{CTS}$, $\overline{DSR}$, $\overline{DCD}$, $\overline{RI}$) are disconnected.
5. The four modem control bits (RTS, DTR, OUT_1, OUT_2) in MCR are internally connected to the four modem status inputs in the MSR (RTS to CTS, DTR to DSR, OUT1 to RI, OUT2 to DCD).

In loop back mode, transmitted data is immediately received. All interrupts are functional and controlled by the Interrupt Enable Register (IER).

Line Status Register (LSR)

The Line Status Register (LSR) provides transfer status information to the ARM7DI. The contents of the register are described in Table 7-10.

Table 7-10.
LSR Fields

LSR BIT POSITION	NAME	DESCRIPTION
0	Data Ready (DR)	Set to 1 when the Receiver Buffer Register is full. Cleared to 0 by reading Receiver Buffer Register.
1	Overrun Error (OE)	Set to 1 if the data in the Receiver Buffer Register was not read before new data from the Receiver Shift Register over wrote it. Cleared to 0 by reading Line Status Register.
2	PARITY Error (PE)	Set to 1 when the received data does not have a valid PARITY. Cleared to 0 by reading Line Status Register.
3	Frame Error (FE)	Set to 1 when the received data does not contain a valid STOP bit. Cleared to 0 by reading the Line Status Register.
4	Break Interrupt (BI)	Set to 1 when the received data is a logic 0 for more than a data trans-mission time (START bit + DATA bits + PARITY + STOP bits). Cleared to 0 by reading Line Status Register.
5	Transmitter Holding Register Empty (THRE)	Set to 1 when Transmitter Holding Register becomes empty, and can accept new data. Cleared to 0 by writing data into Transmitter Holding Register.
6	Transmitter Empty (TEMT)	Set to 1 when Transmitter Shift Register and the Transmitter Holding Register are both empty. Cleared to 0 by writing data into Transmitter Holding Register.
7	—	Always 0

NOTE: LSR is a read only register.

Modem Status Register (MSR)

The Modem Status Register (MSR) provides modem status information. The contents of the register are described in Table 7-11.

Table 7-11.
MSR Fields

MS BIT POSITION	NAME	DESCRIPTION
0	Delta Clear to Send (DCTS)	Set to 1 if the CTS bit has changed since this register was last read.
1	Delta Data Set Ready (DDSR)	Set to 1 if the DSR bit has changed since this register was last read.
2	Trailing Edge Ring Indicator (TERI)	Set to 1 if the $\overline{RI}$ signal has made a LOW to HIGH transition since it was last read.
3	Delta Data Carrier Detect (DDCD)	Set to 1 if the DSR bit has changed since this register was last read.
4	Clear to Send (CTS)	The status of this bit depends on the loop bit in the MCR: **LOOP CTS** 0 Reflects the complement of the external $\overline{CTS}$ input 1 Reflects the value of RTS bit in MCR
5	Data Set Ready (DSR)	The status of this bit depends on the loop bit in the MCR: **LOOP DSR** 0 Reflects the complement of the external $\overline{DSR}$ input 1 Reflects the value of DTR bit in MCR
6	Ring Indicator (RI)	The status of this bit depends on the loop bit in the MCR: **LOOP RI** 0 Reflects the Complement of the external $\overline{RI}$ input 1 Reflects the value of OUT1 bit in MCR
7	Data Carrier Detect (DCD)	The status of this bit depends on the loop bit in the MCR: **LOOP DCD** 0 Reflects the Complement of the external $\overline{DCD}$ input 1 Reflects the value of OUT2 bit in MCR

Bits 0 - 3 can be written to, and a modem status interrupt can be cleared by writing a logic 0 or set by writing a logic 1 to the appropriate bit. Bits 0 - 3 are all reset if the MSR is read. Bits 4 - 7 are input bits.

Scratch Pad Register (SCR)

The Scratch Pad Register (SCR) is a general purpose read/write register.

Divisor Latches (DLL, DLM) – Read/Write

The two divisor latches are used to control the UART baud rate. The divisor is comprised of the DLM and DLL registers, where:

Divisor = (DLM x 256) + DLL
$\overline{BAUD}$ = f(UART_in_CLK)/Divisor
Baud Rate = $\overline{BAUD}$ /16

Solving for the divisor will yield: Divisor = f(UART_in_CLK)/(16 x Baud Rate)

The baud generator will take the input clock to the UART and divide it by the value held in DLM and DLL to produce $\overline{BAUD}$ which is 16 × the desired baud rate. The divisor latches must be initialized after reset of the UART to ensure proper operation of the baud generator. They can only be accessed if DLAB bit is set to 1. When either DLL or DLM is written, the 16-bit Baud counter is loaded immediately and the divisor count is reset. The input clock to the UART can be divided any where from 1 to 2^{16}. Table 7-12 details how $\overline{BAUD}$ is generated.

Table 7-12.
$\overline{BAUD}$ Generation

DIVISOR	$\overline{BAUD}$
1	Same as UART_in_CLK
2	UART_in_CLK/2
3	UART_in_CLK/3
:	:
2^{16} - 1	UART_in_CLK/ 65535
0	UART_in_CLK/ 65536

Table 7-13 shows required divisor to generate a given baud rate from UART_in_CLK input of 1.8432 MHz, 3.072 MHz, and 8 MHz:

Table 7-13.
Baud Rate Division

BAUD RATE	1.8432 MHz	3.072 MHz	8 MHz
50	2304	3840	10000
75	1536	2560	6667*
110	1047*	1745*	4545*
134.5	857*	1428*	3717*
150	768	1280	3333*
300	384	640	1667*
600	192	320	833*
1200	96	160	417*
1800	64	107*	277*
2000	58*	96	250
2400	48	80	208*
3600	32	53*	139*
4800	24	40	104*
7200	16	27*	69*
9600	12	20	52*
19200	6	10	26*
38400	3	5	13*
56000	2*	**	9*
57600	2	**	**
115200	1	**	**
128000	**	**	4*
256000	**	**	2*

NOTES:
1. *Baud rate produced is not the exact baud rate.
2. ** No suitable divisor.

Some of the above divisors do not produce the exact baud rate; for example, running with an 8 MHz UART input clock and a divisor of 4545 produces a baud rate of 110.011 instead of 110, giving an error rate of .01% ((110.011-110)/110).

UARTS INPUT CLOCK

Each UART has its own input clock, UART_in_CLK, which is a scaled down system clock (XCLK) or an external clock provided by UCLK input pin. Selecting between the two clocks is controlled by PCSR register in Clock and Power Management (CPMU) unit and scaling down XCLK is controlled by U0CCR, U1CCR, and U2CCR registers in the CPMU unit.

Chapter 8

SERIAL INFRARED (SIR) INTERFACE

INTRODUCTION

The 790A has a Serial InfraRed (SIR) interface, that is IrDA-1.0 and SHARP-DASK compatible. The interface connects to UART2 and performs Format Encoding/ Decoding between UART format and IrDA/DASK SIR formats. The SIR interfaces directly to any external IrDA/DASK Transceiver module as shown in Figure 8-1.

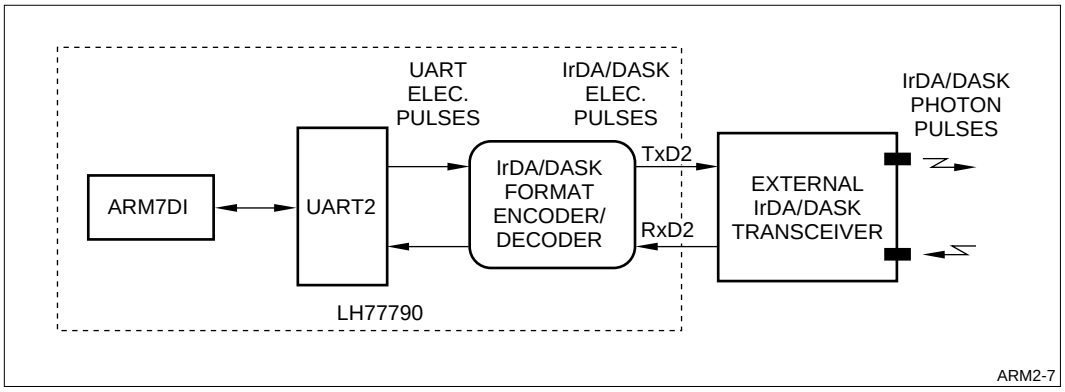


Figure 8-1. SIR Interface Example

FEATURES

- IrDA SIR (Version 1.0) Compatible
- SHARP DASK SIR Compatible
- Adds IR Port to UART2
- 2.4 kbps to 115.2 kbps IrDA Data Rate
- 2.4 kbps to 57.6 kbps DASK Data Rate
- Sleep Mode to Save Power
- Three Modes of Operations:
  - UART: 3 Serial Ports
  - IrDA SIR: 2 Serial Ports + 1 InfraRed Port
  - DASK SIR: 2 Serial Ports + 1 InfraRed Port
- UART Format ↔ IrDA/DASK Format

BLOCK DIAGRAM

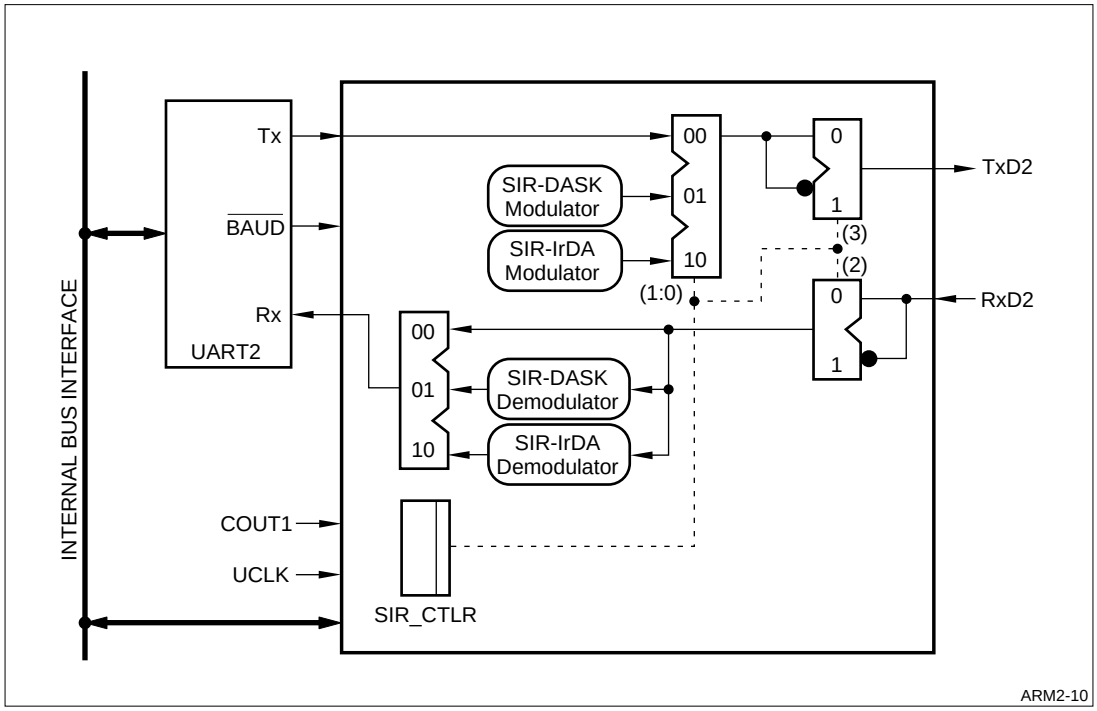


Figure 8-2. SIR Block Diagram

EXTERNAL INTERFACE

Table 8-1.
SIR External Interface

SIGNAL	BITS	INPUT/OUTPUT	FUNCTION
TxD2	1	Output	Transmitter Output for SIR/UART2.
RxD2	1	Input	Receiver Input for SIR/UART2.
UCLK	1	Input	UART/DASK External Clock Input.

REGISTER MAP

The base address for SIR register(s) is 0xFFFF0C00. The offset, initial value, access and number of bits for each register are as follows:

Table 8-2.
SIR Register Offset

SIR REGISTER	ADDRESS OFFSET [7:0]	RESET VALUE	ACCESS	NUMBER OF BITS
SIR_CTLR	0x00	0x02	R/W	8

GENERAL OPERATION

The SIR interface supports two infrared standards: IrDA-1.0 and SHARP DASK. Both standards use photon pulses to send and receive information serially via UART2.

IrDA Mode

IrDA operates at baud rates from 2400 bps to 115.2 kbps. A logical '0' is represented by an IR pulse and a logical '1' is represented by the absence of an IR pulse (no-pulse).

On the transmitter side, the IrDA modulator takes the transmitted output of UART2, and changes a ‘0’ input to a positive pulse of width (3/16) x T (T is: 1/UART2_BaudRate) and changes a ‘1’ input to no-pulse (zero level) as shown in Figure 8-3a. The modulated signal can either be transmitted on TxD2 as is (for a positive pulse), or inverted (for a negative pulse) then transmitted.

On the receiver side, the received data on RxD2 is either inverted (if it is a negative pulse) or passed unchanged to the IrDA demodulator. The demodulator changes a positive pulse to a ‘0’ output for a period of T and changes a no-pulse to a ‘1’ output for a period of T as shown in Figure 8-3b. The demodulated output is then received by UART2.

Since IrDA 1.0 specification only define the optical signals, the inversion on the output of modulator and the input of the demodulator were added to allow the 790A to interface to a wide range of IrDA transceivers. SIR_CTLR controls the inversion of TxD2 and RxD2.

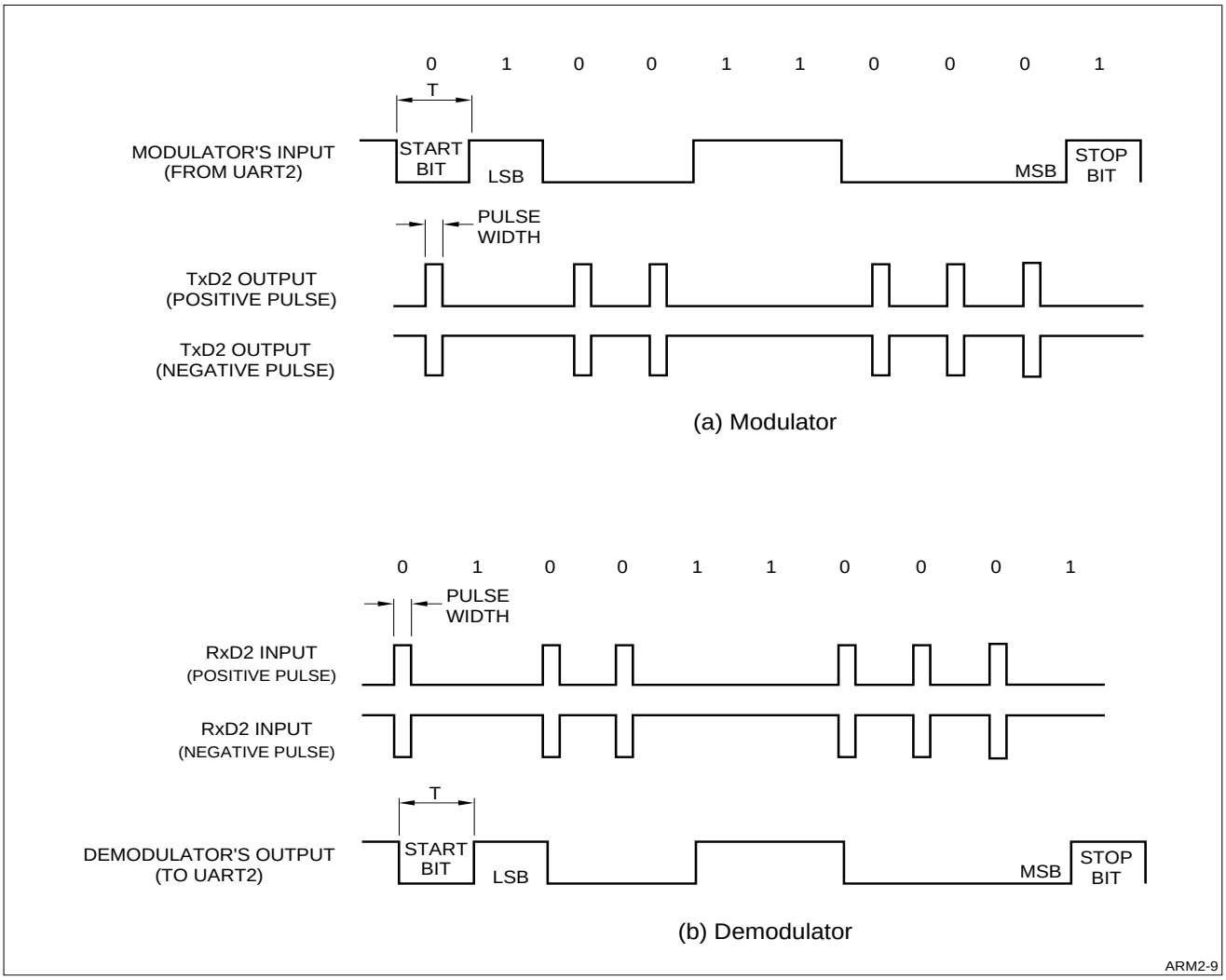


Figure 8-3. IrDA Mode

DASK Mode

Sharp's Digital implementation for Amplitude Shift Keying (DASK) offers infrared communication link at bit rate up to 57.6 kbps with ASK modulation.

A logical '0' is indicated by the presence of a 50% duty 500 kHz infrared carrier wave. A logical '1' is indicated by the absence of such carrier wave.

On the transmitter side, the DASK modulator takes the transmitted output of UART2, inverts it and then logically ANDs it with a 500 kHz, 50% duty clock from counter/timer1. The modulated signal is then transmitted on TxD2 (Figure 8-4).

On the receiver side, the demodulator detects a high to low transition on RxD2 input and demodulates the received data to UART format by passing it through a digital band pass filter that operates at 14.318 MHz external clock. The demodulated output is then received by UART2 (Figure 8-5).

UART Mode

In this mode of operation, the SIR interface simply passes the transmitted data from UART2 to TxD2 and the received data from RxD2 to UART2. This allows UART2 to function as a standard UART.

Both the transmitted and received data can be also inverted if desired.

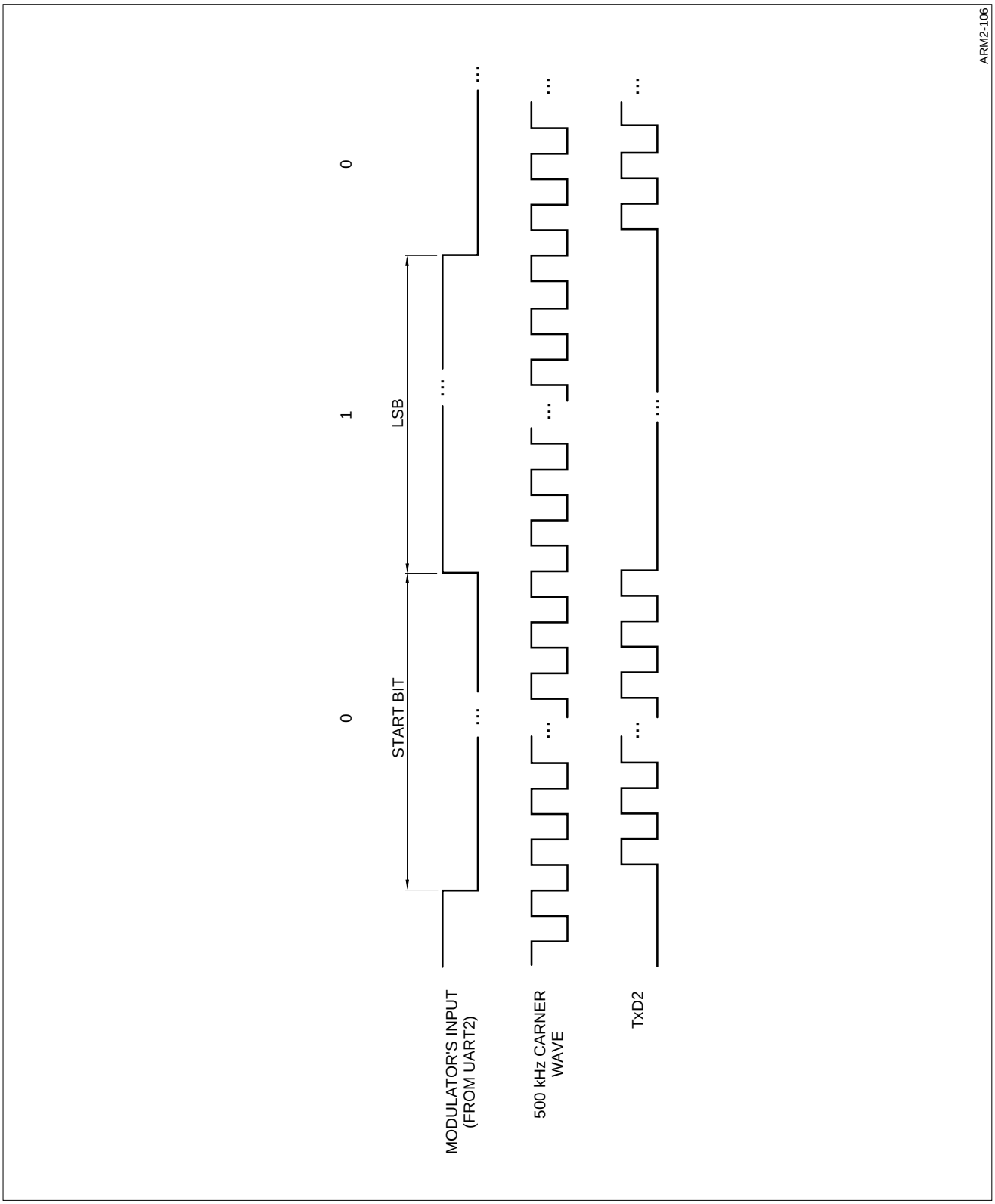


Figure 8-4. DASK Mode: Modulator

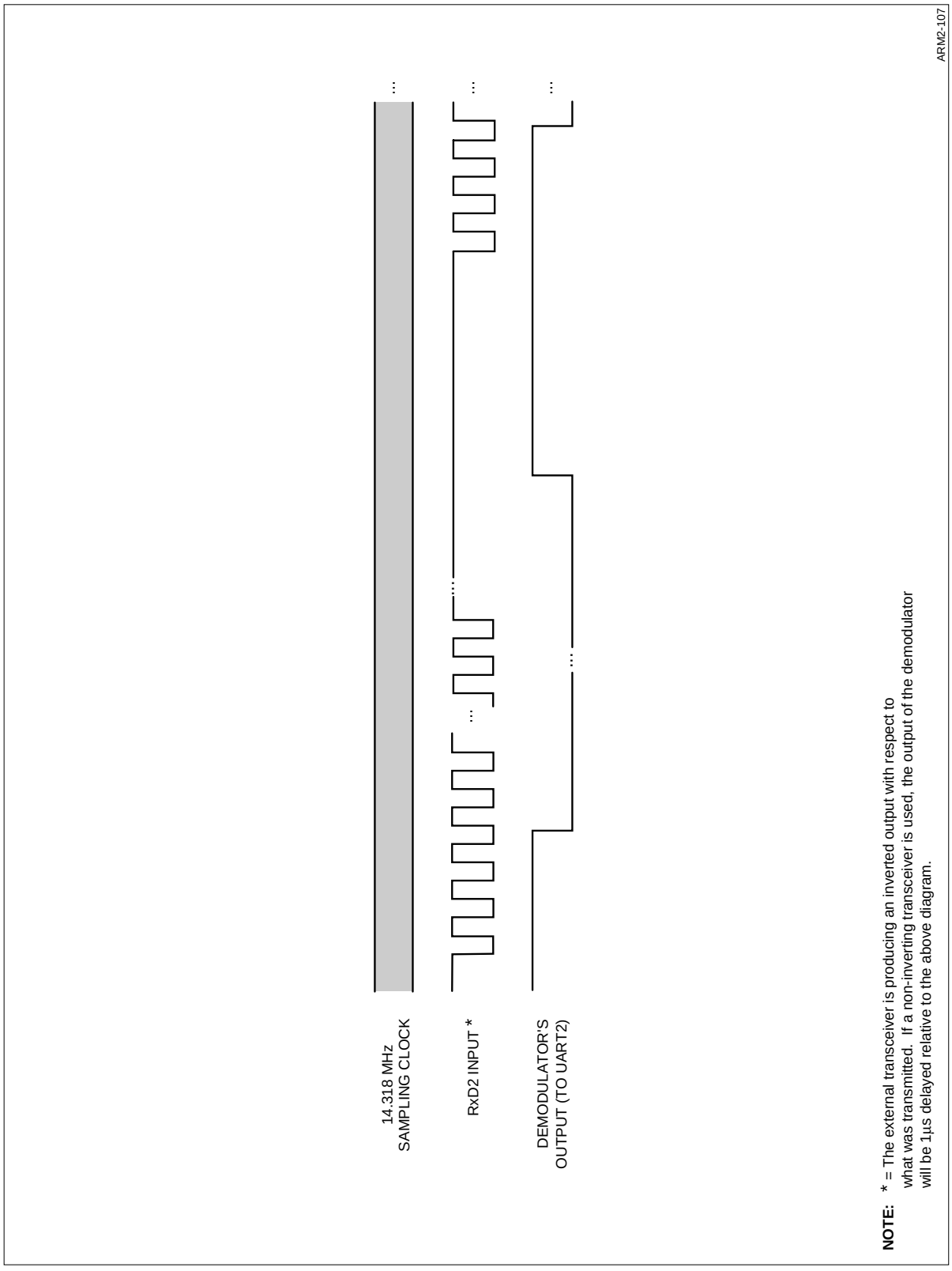
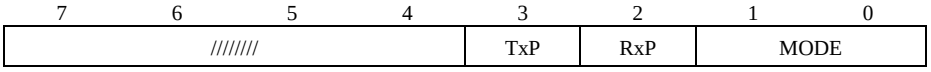


Figure 8-5. DASK Mode: Demodulator

REGISTER DESCRIPTION

SIR Control Register (SIR_CTLR)



The SIR has an 8-bit control register to control its functionality as follows:

Table 8-3.
SIR_CTLR Fields

BIT POSITION	NAME	DESCRIPTION
7:4	Reserved	Reserved.
3	TxP	Tx Polarity (UART and IrDA modes only) 0 ---> TxD2 output will not be inverted before it is sent out. 1 ---> TxD2 output will be inverted before it is sent out.
2	RxP	Rx Polarity (UART & IrDA modes only) 0 ---> RxD2 input will not be inverted before processing it. 1 ---> RxD2 input will be inverted before processing it.
1:0	Mode	UART2 Operation Mode 00 ---> UART Mode 01 ---> DASK Mode 10 ---> IrDA Mode 11 ---> Illegal.

IRDA CLOCK

The IrDA input clock is the same as the baud rate generator output clock for UART2, $\overline{\text{BAUD}}$. It runs 16x baud rate. This clock will be used to sample data 16 times per period.

DASK CLOCKS

The DASK requires a 500 kHz IR carrier wave with 50% duty cycle. Counter/Timer1, programmed as a square ware generator, must be used to generate the 500 kHz clock. The output of the Counter/Timer, CTOUT1, is connected internally to the DASK Modulator.

The DASK also requires a 14.318 MHz for its Demodulator module. This clock must come from an external oscillator that should be connected to UCLK input pin. UCLK is the external UART clock, and is internally connected to the DASK Demodulator.

POWER MANAGEMENT MODE

All of the SIR Clocks can be disabled by SIRCE bit in PCSR register in the Clock and Power Management Unit (CPMU) to save power. Refer to CPMU chapter for more detail.

Chapter 9

PULSE WIDTH MODULATOR (PWM)

INTRODUCTION

The 790A supports three fully independent programmable Pulse Width Modulator channels (PWM0, PWM1, PWM2) with a frequency range shown in Table 9-1.

Table 9-1.
PWM Frequency Range

SYSTEM CLOCK FREQUENCY	PWM PULSE FREQUENCY RANGE	
	8-BIT RESOLUTION (PWM0 AND PWM1)	16-BIT RESOLUTION (PWM2)
16.7 MHz	1.05 kHz – 4.17 MHz or DC	4.10 Hz – 4.17 MHz or DC
25 MHz	1.58 kHz – 6.25 MHz or DC	6.15 Hz – 6.25 MHz or DC

PWM DEFINITION

Pulse Width Modulation (PWM) is simply a method of communicating information to a device. Figure 9-1 shows an example of a PWM output signal derived from the system clock.

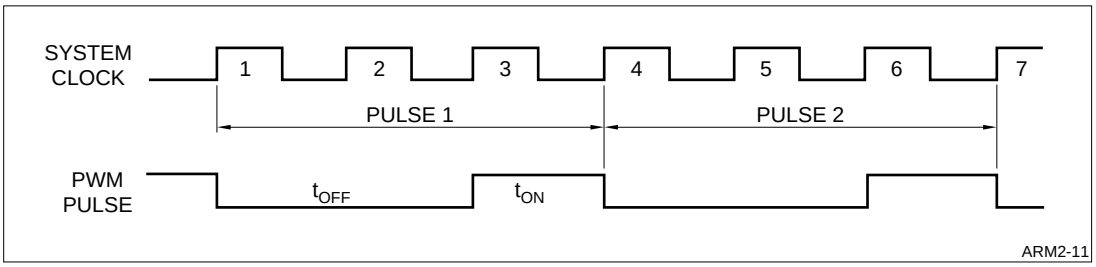


Figure 9-1. PWM Timing Diagram

The PWM period and the duty cycle are programmable. The duty cycle is expressed as the duration of t_{ON} over the sum of t_{ON} and t_{OFF}. A signal has a constant duty cycle if t_{ON} and t_{OFF} are uniform. If t_{ON} is equal to t_{OFF}, the signal has a 50% duty cycle.

$$\text{Duty Cycle} = t_{ON}/(t_{ON} + t_{OFF})$$

FEATURES

- Each PWM Channel is Independent
- Frequency Range:
  - DC to 6.25 MHz
- Easy to Program
- PWM0 and PWM1 has 8-Bit Resolution
- PWM2 has 16-bit Resolution
- Programmable Synchronous Mode Support
  - Allows on-chip Counter/Timer0 to start PWMs
  - This Produces a Synchronized Sequence of PWM Pulses
- Programmable Pulse Width (Duty Cycle), Interval (Frequency) and Polarity
  - Static Programming: PWM is Stopped
  - Dynamic Programming: PWM is Running
    - Double Buffering Allows Dynamic Programming
    - Updates to Duty Cycle, Frequency, and Polarity are Done at End of a PWM Cycle
- Start/Stop PWM
  - Stop PWM Output at End of a PWM Cycle
- Sleep Mode to Save Power
- PWM Output Connected to On-Chip Counters
  - This Feature Simplifies Building Applications like A/D

PWM APPLICATIONS

Pulse Width Modulation (PWM) is used in a variety of applications, such as:

- LED Intensity Control
- LCD Gain and Contrast
- Automotive Engine Control
- DC Motor Speed
- D/A and A/D conversion
- Sound Synthesis
- Laser Applications
- Servo Motor Control

FUNCTIONAL BLOCK DIAGRAM

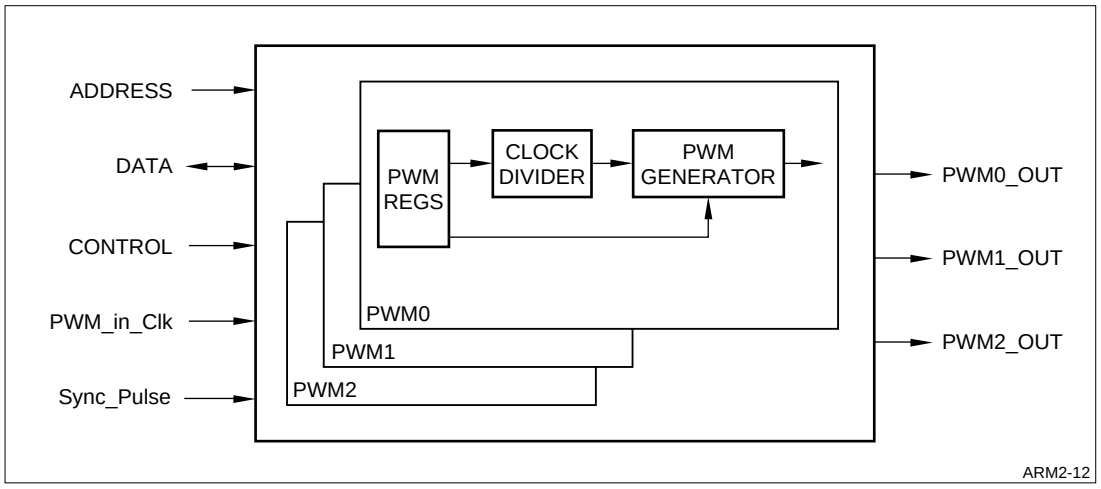


Figure 9-2. Functional Block Diagram

EXTERNAL INTERFACE

Table 9-2.
PWM External Interface

SIGNAL	BIT	INPUT/OUTPUT	FUNCTION
PWM0_OUT PWM1_OUT PWM2_OUT	3	Output	Outputs of PWM0, PWM1, and PWM2. They map to 790A output pins PWM0, PWM1 and PWM2 respectively

REGISTER MAP

The base address for PWM registers is 0xFFFF1000. The offset, initial value, access and number of bits for each register are as follows:

Table 9-3.
PWMs' Register Map

PWM REGISTER	ADDRESS OFFSET [7:0]	RESET VALUE	ACCESS	NUMBER OF BITS
PWM0_TC[1]	00	0x00	W	8
PWM0_DC[1]	04	0x00	W	8
PWM0_ENB	08	0x00	W	8
PWM0_DIV[1]	0C	0x00	W	8
PWM0_SYNC[2]	10	0x00	W	8
PWM0_INV	14	0x00	W	8
PWM0_UPDT[2]	18	0x00	W	8
PWM1_TC[1]	20	0x00	W	8
PWM1_DC[1]	24	0x00	W	8
PWM1_ENB	28	0x00	W	8
PWM1_DIV[1]	2C	0x00	W	8
PWM1_SYNC[2]	30	0x00	W	8
PWM1_INV	34	0x00	W	8
PWM1_UPDT[2]	38	0x00	W	8
PWM2_TC[1]	40	0x0000	W	16
PWM2_DC[1]	44	0x0000	W	16
PWM2_ENB	48	0x00	W	8
PWM2_DIV[1]	4C	0x00	W	8
PWM2_SYNC[2]	50	0x00	W	8
PWM2_INV	54	0x00	W	8
PWM2_UPDT[2]	58	0x00	W	8
PWMA_TC[1]	60	0x0000	W	16
PWMA_DC[1]	64	0x0000	W	16
PWMA_ENB	68	0x00	W	8
PWMA_DIV[1]	6C	0x00	W	8
PWMA_SYNC[2]	70	0x00	W	8
PWMA_INV	74	0x00	W	8
PWMA_UPDT[2]	78	0x00	W	8

NOTES:

1. Reset value is ILLEGAL
2. PWM must be stopped to write to this register.

GENERAL OPERATION

All three PWMs are identical except for the fact that PWM0 and PWM1 have 8-bit resolution and PWM2 has 16-bit resolution. Each PWMn (n = 0, 1, 2) can function in either normal mode or synchronous mode as follows:

Normal Mode

PWMn operates in normal mode when PWMn_SYNC register is reset to '0'. PWMn will be independently enabled and started via writing a '1' to register PWMn_ENB. All PWMs are set to this mode upon reset. A PWMn can be stopped by writing a '0' to PWMn_ENB. It will stop at the end of the current PWM cycle.

Example:

Figure 9-3 shows PMW0 programmed in Normal Mode.

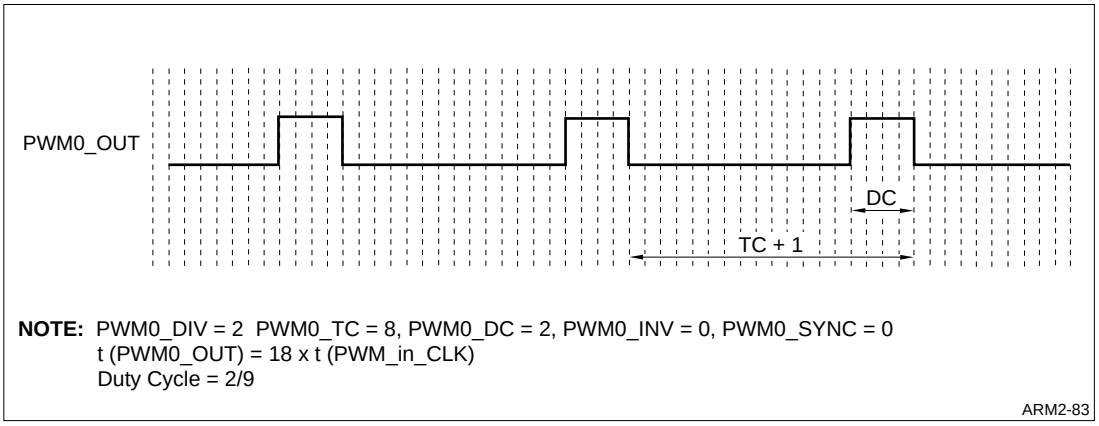


Figure 9-3. PWM0 Running in Normal Mode

Synchronous Mode

PWMn operates in synchronous mode when both PWMn_ENB and PWMn_SYNC register are set to '1'. In this mode, the PWMs will be started synchronously and sequentially by the on-chip Counter/Timer0 output (CTOUT0). Counter/Timer0 should be programmed to operate as a Rate Generator (See the Counters/Timers chapter). Counter/Timer0 will produce a pulse on its output signal, every N Counter/Timer0 clock cycles (N = Counter Value, $2 \leq N \leq 2^{16} - 1$). Any of the PWMs running in synchronous mode can be stopped by writing a '0' to PWMn_ENB. It will stop at the end of the current PWM cycle. The stopped PWM can then be used in normal mode by reseting its PWMn_SYNC to '0'.

Table 9-4.
Synchronous Mode

PWM0_SYNC	PWM1_SYNC	PWM2_SYNC	PWMs IN SYNCHRONOUS MODE (IN STARTING ORDER)	PWMs IN NORMAL MODE
0	0	0	None	None
0	0	1	PWM2	PWM0
0	1	0	PWM1	PWM0, PWM2
0	1	1	PWM1, PWM2	PWM0
1	0	0	PWM0	PWM1, PWM2
1	0	1	PWM0, PWM2	PWM1
1	1	0	PWM0, PWM1	PWM2
1	1	1	PWM0, PWM1, PWM2	None

Example:

If all three PWMs are programmed to operate in synchronous mode and Counter/Timer0 is programmed as a Rate Generator, the following sequence of events will take place (Figure 9-4):

1. PWM0 will be started by the first clock pulse of Counter/Timer0 output.
2. PWM1 will be started by the second clock pulse of Counter/Timer0 output.
3. PWM2 will be started by the third clock pulse of Counter/Timer0 output.

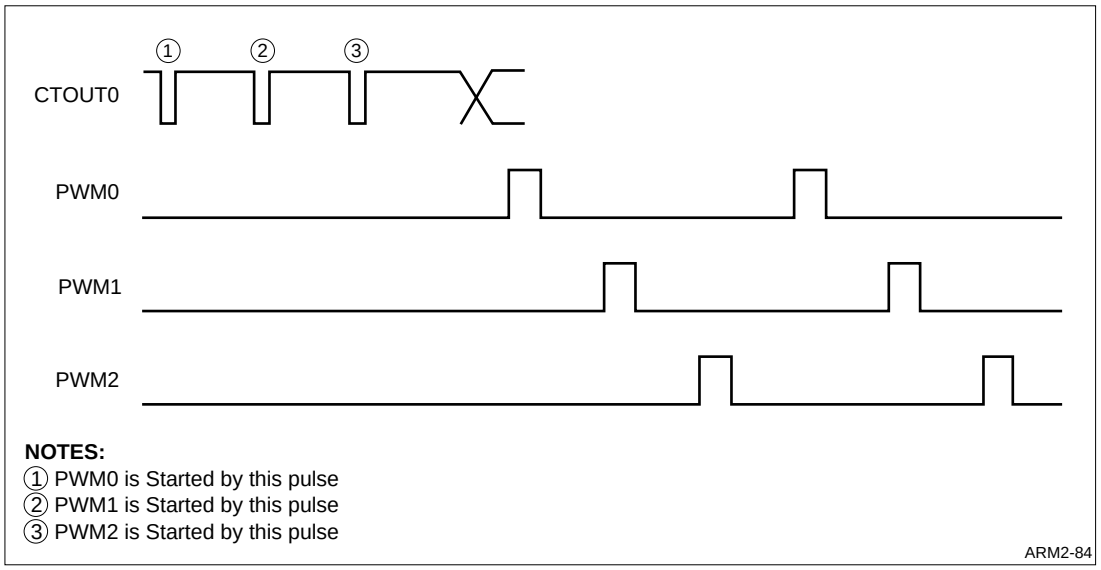


Figure 9-4. PWM0, PWM1, PWM2 Running in Synchronous Mode

NOTE: Once all PWMs are started synchronously, the Counter/Timer0 will no longer be needed. The only difference between a PWM running in synchronous mode and another running in normal mode is the way it is started.

POWER MANAGEMENT

All three PWM channels share the same input clock, PWM_in_CLK, from the Clock and Power Management Unit (CPMU). This input clock is equal to XCLK once enabled by CPMU.

All three PWM channels can be turned off via the Power Management Unit to save power. This will turn off the input clock, PWM_in_CLK. The PWM output(s) will be frozen, either 1 or 0 depending on the state of the PWM output. To force a specific value:

1. Stop/Disable all PWMs
2. Select Inverted or Non-Inverted output
3. Update PWMs
4. Put PWMs in Sleep mode

The above sequence will give you a DC PWM pulse (0 or 1) and save power at the same time.

STATUS DURING RESET

The PWMs will drive a logical '0' during Reset. Coming out of Reset, it will continue to drive a logical '0', and the PWMs will be in halt mode.

PWM PULSE FREQUENCY AND DUTY CYCLE

PWMn (n = 0, 1, 2) output pulse frequency is dependent on PWMn_DIV and PWMn_TC values:

$$t_{PWMn\_OUT} = PWMn\_DIV \times (PWMn\_TC + 1) \times t_{PWM\ \_in\_CLK}$$

$$1 \leq PWMn\_TC \leq 2^{16} - 1$$

$2 \leq PWMn\_DIV \leq 62$, PWMn_DIV is an even number

PWMn Duty Cycle is dependent on PWMn_TC and PWMn_DC values only:

$$Duty\ Cycle = (PWMn\_DC)/(\ PWM\_TC + 1)$$

$$1 \leq PWMn\_DC \leq PWMn\_TC$$

NOTE: PWMn_TC, PWMn_DC, and PWMn_DIV values that do not meet the above limits are illegal and the output is not guaranteed.

Solving for PWMn_TC and PWMn_DC will result in (with the above limits):

$$PWMn\_TC = [t_{PWMn\_OUT}/(PWMn\_DIV \times t_{PWM\_in\_CLK})] - 1$$

$$PWMn\_DC = Duty\ Cycle \times (PWMn\_TC + 1)$$

Example:

To produce a PWM0 output of 500 kHz (2 µs) and 20% duty cycle with a system clock of 25 MHz (40 ns):

1. Select a Divide value of 10: PWM0_DIV = 10 (decimal)

2. Calculate $PWM0\_TC = [2\ \mu s/(40\ ns \times 10)] - 1 = 4$ (decimal)

3. Calculate $PWM0\_DC = 0.2 \times (4 + 1) = 1$ (decimal)

REGISTER DESCRIPTION

Clock Divider Registers [PWMn_DIV (n = 0, 1, 2)]

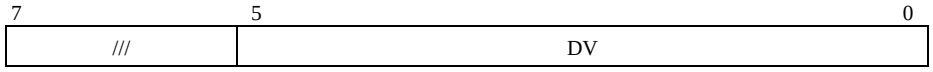


Table 9-5.
PWMn_DIV Fields

BIT POSITION	NAME	DESCRIPTION
5:0	DV	Divide Value: This is used to divide the PWM input clock, PWM_in_CLK, by any even value in the range [2:62], Bit 0 is always treated as 0: **DV PWMn_Clk** 00000X ----> Illegal 00001X ----> PWM_in_CLK/2 00010X ----> PWM_in_CLK/4 00011X ----> PWM_in_CLK/6 00100X ----> PWM_in_CLK/8 : 11111X ----> PWM_in_CLK/62 The new divided clock, PWMn_Clk, will drive the logic to produce a PWMn output. PWM_in_CLK is a gated XCLK Clock (See 'Clock and Power Management' Chapter)
7	///	Reserved

NOTE: X = Don't care

PWMn_DIV is used in conjunction with PWMn_TC to adjust the output frequency of PWMn. PWMn_DIV is double buffered to allow it to be programmed statically (PWMn is stopped) or dynamically (PWMn is running). Programmed dynamically, PWMn_DIV is updated at the end of a PWMn cycle thus causing no output glitches or errors.

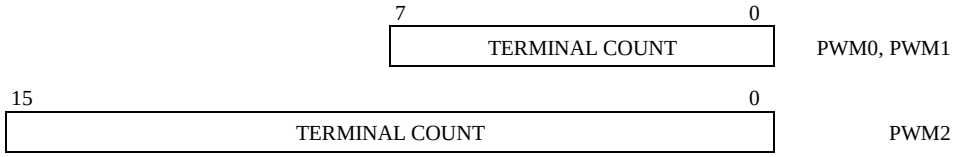
A read/write access to PWMn_DIV will read/write its buffer. The buffered data will be written to PWMn_DIV as follows:

Table 9-6.
PWMn_DIV Update

PWMN STATE	PWMn_DIV UPDATED
Stopped	A write to PWMn_UPDT
Running	At the End of a PWMn_Cycle

A write to PWMA_DIV allows writing a common divide value to PWM0_DIV, PWM1_DIV and PWM2_DIV at the same time.

Terminal Count Registers [PWMn_TC (n = 0, 1, 2)]



PWMn_TC is an 8-bit Terminal Count for PWM0 and PWM1, and a 16-bit Terminal Count for PWM2. It is used in conjunction with PWMn_DIV to adjust the output frequency of PWMn. PWMn_TC gives PWM0 and PWM1 an 8-bit resolution and PWM2 a 16-bit Resolution. PWMn_TC is double buffered to allow it to be programmed statically (PWMn is stopped) or dynamically (PWMn is running). Programmed dynamically, PWMn_TC is updated at the end of a PWMn cycle, thus causing no output glitches or errors.

A read/write access to PWMn_TC will read/write its buffer. The buffered data will be written to PWMn_TC as follows:

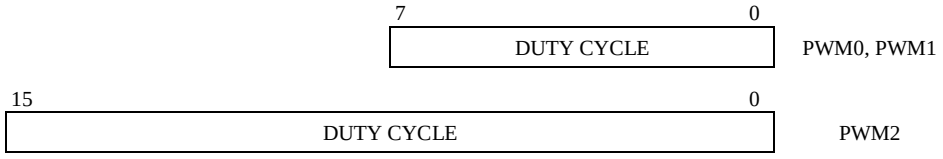
Table 9-7.
PWMn_TC Update

PWMN STATE	PWMN_TC UPDATED
Stopped	A write to PWMn_UPDT
Running	At the End of a PWMn_Cycle

A write to PWMA_TC allows writing a common TC value to PWM0_TC, PWM1_TC, and PWM2_TC at the same time. PWM0_TC and PWM1_TC will get the low order 8-bits and PWM2_TC will get the low order 16-bits.

Refer to 'PWM Pulse Frequency and Duty Cycle' section for restrictions on PWMn_TC.

Duty Cycle Registers [PWMn_DC (n = 0, 1, 2)]



PWMn_DC is an 8-bit duty cycle for PWM0 and PWM1, and a 16-bit duty cycle for PWM2. It is used in conjunction with PWMn_TC to adjust the output duty cycle of PWMn. PWMn_DC gives PWM0 and PWM1 an 8-bit resolution and PWM2 a 16-bit Resolution. PWMn_DC is double buffered to allow it to be programmed statically (PWMn is stopped) or dynamically (PWMn is running). Programmed dynamically, PWMn_DC is updated at the end of a PWMn cycle thus causing no output glitches or errors.

A read/write access to PWMn_DC will read/write its buffer. The buffered data will be written to PWMn_DC as follows:

Table 9-8.
PWMn_DC Update

PWMn STATE	PWMn_DC UPDATED
Stopped	A write to PWMn_UPDT
Running	At the End of a PWMn_Cycle

A write to PWMA_DC allows writing a common DC value to PWM0_DC, PWM1_DC, and PWM2_DC at the same time. PWM0_DC and PWM1_DC will get the low order 8-bits and PWM2_DC will get the low order 16-bits.

Refer to 'PWM Pulse Frequency and Duty Cycle' section for restrictions on PWMn_DC.

Output Invert Registers [PWMn_INV (n = 0, 1, 2)]

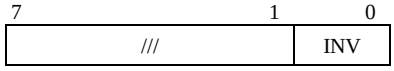


Table 9-9.
PWMn_INV Fields

BIT POSITION	NAME	DESCRIPTION
0	INV	Invert PWMn Output: 0: Output is not inverted. PWMn_OUT will output T_{OFF} first then t_{ON}. PWMn_DC controls t_{ON}. 1: Output is inverted. PWMn_OUT will output t_{ON} first then t_{OFF}. PWMn_DC controls t_{OFF}.
7:1	///	Reserved

PWMn_INV is double buffered to allow it to be programmed statically (PWMn is stopped) or dynamically (PWMn is running). Programmed dynamically, PWMn_INV is updated at the end of a PWMn cycle thus causing no output glitches or errors.

A read/write access to PWMn_INV will read/write its buffer. The buffered data will be written to PWMn_INV as follows.

Table 9-10.
PWMn_INV Update

PWMn STATE	PWMn_INV UPDATED
Stopped	A write to PWMn_UPDT
Running	At the End of a PWMn_Cycle

This will allow the system designer to program the PWMn output to fit any application. The following diagram provides an example of two identical but inverted output pulses.

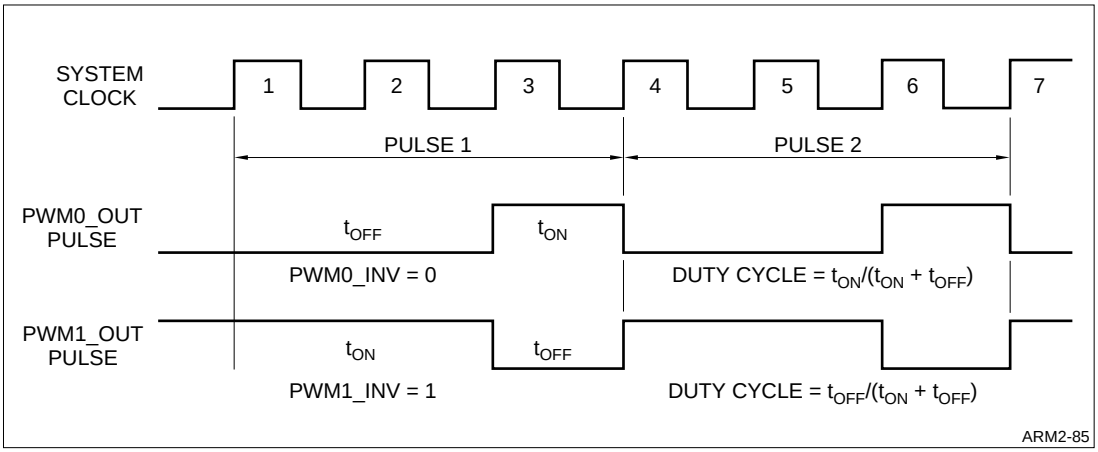


Figure 9-5. PWMn_INV Example

A write to PWMA_INV allows writing a common value to PWM0_INV, PWM1_INV, and PWM2_INV at the same time.

Enable Registers [PWMn_ENB (n = 0, 1, 2)]

7 ... 1	0
///	ENB

Table 9-11.
PWMn_ENB Fields

BIT POSITION	NAME	DESCRIPTION
0	ENB	Enable PWMn. 0: Disable/Stop PWMn. A running PWM is actually stopped when it reaches the end of its cycle in Normal and Synchronous modes. PWMn output will be: 0 ---> If PWMn_INV[0] = 0 1 ---> if PWMn_INV[0] = 1 1: PWMn is Enabled. If PWMn is in Normal Mode, it will *start running on the next cycle.* If PWMn is in Synchronous Mode, it is enabled to *start running on the next cycle after the appropriate counter pulse* as explained in the Synchronous Mode section.
7:1	///	Reserved

A write to PWMA_ENB allows writing a common value to PMW0_ENB, PWM1-ENB, and PWM2_ENB at the same time.

Synchronous Registers [PWMn_SYNC (n = 0, 1, 2)]

7 ... 1	0
///	SYNC

Table 9-12.
PWMn_SYNC Fields

BIT POSITION	NAME	DESCRIPTION
0	SYNC	PWMn Synchronous mode: 0: PWMn is in Normal mode 1: PWMn is in Synchronous mode
7:1	///	Reserved

A write to PWMA_SYNC allows writing a common value to PWM0_SYNC, PWM1_SYNC, PWM2_SYNC at the same time. PWMn_SYNC and PWMA_SYNC registers can't be programmed dynamically (while the PWM is running).

Update Registers [PWMn_UPDT (n = 0, 1, 2)]

7 ... 1	0
///	UPDT

Table 9-13.
PWMn_UPDT

BIT POSITION	NAME	DESCRIPTION
0	UPDT	Writing to this register will Update a Stopped PWMn as follows: PWMn_TC is updated with its new buffered value. PWMn_DC is updated with its new buffered value. PWMn_DIV is updated with its new buffered value. PWMn_INV is updated with its new buffered value. The value written to this register does not matter.
7:1	///	Reserved

A write to PWMA_UPDT allows updating all PWMs at the same time. PWMn_UPDT and PWMA_UPDT can't be programmed dynamically (while the PWM is running).

PWM CONNECTION TO ON-CHIP COUNTER/TIMERS

Each of the PWMs output can be used as a Gate source for the on-chip counters/timers. This feature is controlled via the I/O Configuration Register (IOCR). This allows the PWMs output to enable, disable, trigger, and synchronize counting. This feature can be used in building an A/D converter using PWM output and external logic.

PWM PROGRAMMING EXAMPLES

(1) Static Programming (PWM is not running)

Assumption:

PWM0 is running in Normal mode (PWM0_SYNC = 0x00, PWM0_ENB = 0x01).

Table 9-14.
Programming Steps

STEP	REGISTER ASSIGNED	VALUE ASSIGNED
Stop PWM0	PWM0_ENB	00000000
Wait for PWM0 to finish current cycle	—	—
Program DIVIDE value with 10	PWM0_DIV	00001010
Program TC value with 4	PWM0_TC	00000100
Program DC value with 1	PWM0_DC	00000001
Program PWM0 output to Invert	PWM0_INV	00000001
Update PWM0	PWM0_UPDT	XXXXXXXX
Enable/Start PWM0	PWM0_ENB	00000001

(2) Dynamic Programming (PWM is running)

Assumption:

PWM0 is running in Normal mode (PWM0_SYNC = 0x00, PWM0_ENB = 0x01).

Table 9-15.
Programming Steps

STEP	REGISTER ASSIGNED	VALUE ASSIGNED
Program DIVIDE value with 30	PWM0_DIV	00011110
Program TC value with 10	PWM0_TC	00001010
Program DC value with 9	PWM0_DC	00001001

NOTE: Updates will take place at the end of the PWM cycle. Order of programming TC and DC are important.

(3) Synchronous Mode Programming

Assumptions:

All PWMs are running in Normal mode (PWMn_SYNC = 0x00, PWMn_ENB = 0x01).

Want PWM0 and PWM2 to run in synchronous mode with same parameters.

Want Counter/Timer0 to generate a pulse every 10 system cycles (XCLK).

Table 9-16.
Programming Steps

STEP	REGISTER ASSIGNED	VALUE ASSIGNED
Stop PWM0	PWM0_ENB	00000000
Stop PWM2	PWM0_ENB	00000000
Wait for PWM0 and PWM2 to finish current cycles	—	—
Program PWM0_DIV value with 10	PWM0_DIV	00001010
Program PWM0_TC value with 4	PWM0_TC	00000100
Program PWM0_DC value with 1	PWM0_DC	00000001
Put PWM0 in Synchronous mode	PWM0_SYNC	00000001
Update PWM0	PWM0_UPDT	XXXXXXXX
Program PWM2_DIV value with 10	PWM2_DIV	00001010
Program PWM2_TC value with 4	PWM2_TC	00000100
Program PWM2_DC value with 1	PWM2_DC	00000001
Put PWM2 in Synchronous mode	PWM2_SYNC	00000001
Update PWM2	PWM2_UPDT	XXXXXXXX
Enable PWM0	PWM0_ENB	00000001
Enable PWM2	PWM2_ENB	00000001
Program Counter/Timer0 with: Count = 5 Rate Generator Mode Counter/Timer0 Clock = $f_{XCLK}/2$	See Counter/Timer section	—
Activate Counter/Timer0	See Counter/Timer section	—

NOTE: The first pulse of Counter/Timer0 will start PWM0 and the second pulse will start PWM2.

(4) Synchronous Mode Programming

Assumptions:

All PWMs are running in Normal mode (PWMn_SYNC = 0x00, PWMn_ENB = 0x01).

Want All PWMs to run in synchronous mode with same parameters and outputs inverted.

Want Counter/Timer0 to generate a pulse every six system cycles (XCLK).

Table 9-17.
Programming Steps

STEP	REGISTER ASSIGNED	VALUE ASSIGNED
Stop all PWMs	PWMA_ENB	00000000
Wait for all PWMs to finish current cycles	—	—
Program all PWMs DIVIDE value with 20	PWMA_DIV	00010100
Program all PWMs TC value with 10	PWMA_TC	0000000000001010
Program all PWMs DC value with 3	PWMA_DC	0000000000000011
Invert all PWMs	PWMA_INV	00000001
Put all PWMs in Synchronous mode	PWMA_SYNC	00000001
Update all PWMs	PWMA_UPDT	XXXXXXXX
Enable all PWMs	PWMA_ENB	00000001
Program Counter/Timer0 with: Count = 3 Rate Generator Mode Counter/Timer0 Clock = f_{XCLK} /2	See Counter/Timer section	—
Activate Counter/Timer0	See Counter/Timer section	—

NOTE: The first pulse of Counter/Timer0 will start PWM0, the second pulse will start PWM1 and the third pulse will start PWM2.

Programming Rules (n = 0, 1, 2, A)

1. While programming PWMn_TC and PWMn_DC on the fly (dynamically), both PWMn_TC and PWMn_DC may not be updated simultaneously at the end of a PWMn cycle. The following rules should be followed to preserve the relationship PWMn_DC ≤ PWMn_TC:

   If PWMn_TC (new) > PWMn_TC (current):
   Program PWMn_TC (new) first then PWMn_DC (new)

   If PWMn_TC (new) < PWMn_TC (current):
   program PWMn_DC (new) first then PWMn_TC (new)

2. Never program PWMn_SYNC or PWMn_UPDT on the fly (dynamically).
3. Always update a stopped PWMn if its parameters (TC, DC, DIV, INV) have changed.
4. Program PWMn_TC and PWMn_DC with values that meet the specification.
5. When a PWM is stopped (PWMn_ENB = 0), it does not stop immediately but waits for the end of the current PWM cycle then stops.

Chapter 10

LCD CONTROLLER

INTRODUCTION

The LCD controller provides control and pixel data for directly driving LCD displays. The video frame buffer resides in the system memory to reduce external memory cost. The LCD controller supports two primary modes, Binary mode (on, off), and Gray mode (on, off, or two gray shades).

FEATURES

- Programmable
- Frame Buffer Resides in Main Memory
- LCD Display Modes
  - Binary Mode: Pixels are ON or OFF (1 bit/pixel)
  - Gray Mode: Pixels are ON, OFF, Gray Shade 1 or Gray Shade 2 (2 bits/pixel)
- LCD Panel Functions
  - Single (4-Bit and 8-Bit Data Transfer)
  - Dual (4-Bit Data Transfer)
  - Panel Division
  - OR Function
- Maximum Resolution
  - Horizontal: 2048 pixels in Binary Mode, 1024 pixels in Gray Mode
  - Vertical: 1024 lines in Single Scan, 2048 lines in Dual Scan
- Supports Virtual Display Screen
- ARM7DI and the LCD controller can run concurrently
- Programmable Bit-Order
- Supports Simple 3-bit/pixel Passive Color Panels

BLOCK DIAGRAM

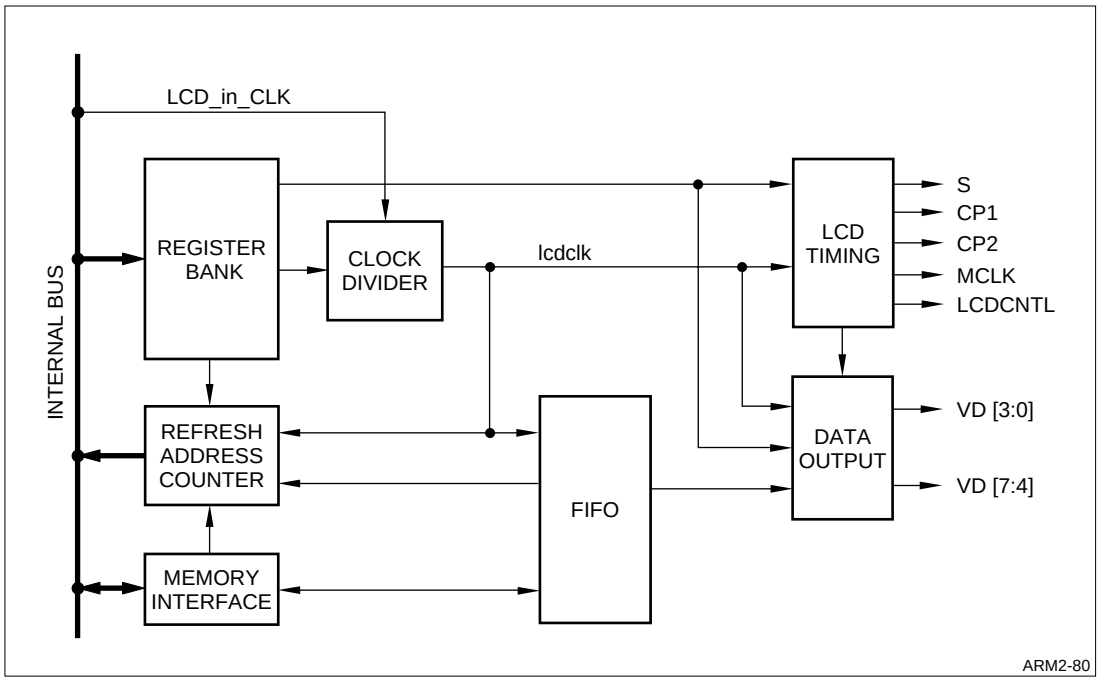


Figure 10-1. Block Diagram

EXTERNAL INTERFACE

Table 10-1.
LCD Panel Interface Signals

NAME	TYPE	DESCRIPTION
VD[3:0]	Output	Video Data for: Single Scan: 4-bit and Lower bits for 8-bit transfers Dual Scan: Upper screen
VD[7:4]	Output	Video Data for: Single Scan: Upper bits for 8-bit transfers Dual Scan: Lower screen
CP2	Output	Shift/Pixel Clock
CP1	Output	Line Pulse/HSYNC
S	Output	Frame Clock/VSYNC
MCLK	Output	AC Modulation Signal
LCDCNTL	Output	LCD Control Signal

LCD PANEL SIGNALS DESCRIPTION

VD[7:0]

These are LCD display data. These data outputs relate to a typical LCD panel as follows:

Table 10-2.
LCD Display Data

LCD CONTROLLER VIDEO OUTPUT PINS	SINGLE SCAN MODE PANELS		DUAL SCAN MODE PANELS
	4-BIT TRANSFER	8-BIT TRANSFER	4-BIT TRANSFER
VD[3:0]	D0:D3	D4:D7	UD0:UD3
VD[7:4]	–	D0:D3	LD0:LD3

D3/UD3/LD3 are the left most bit displayed on the panel for 4-bit transfer. D7 is the left most bit displayed on the panel for 8-bit transfer

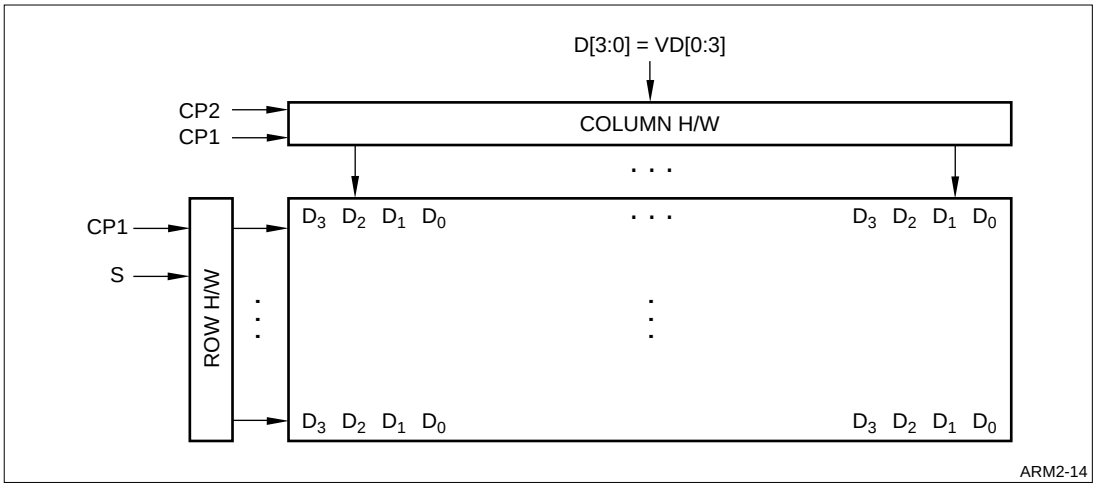


Figure 10-2. Single Scan Mode 4-Bit Transfer

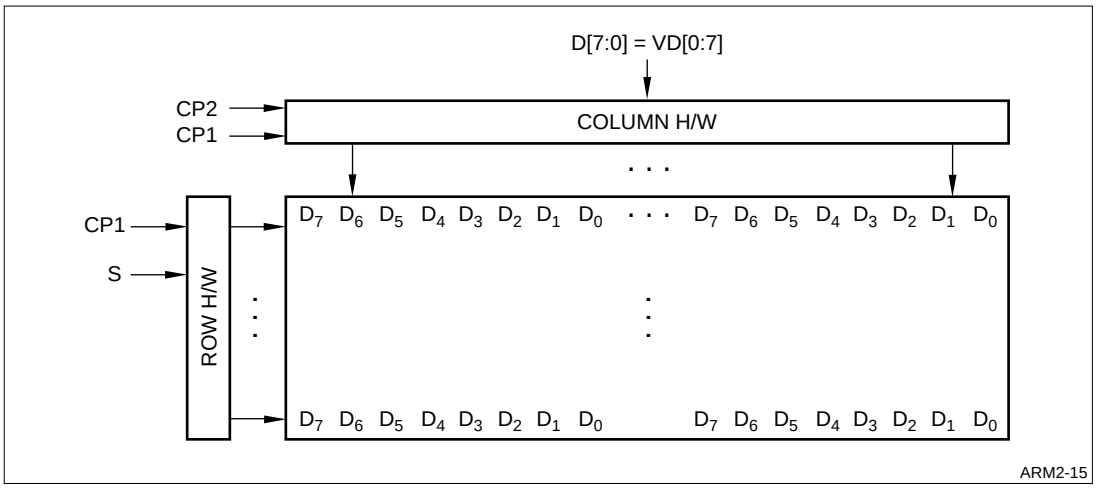


Figure 10-3. Single Scan Mode 8-Bit Transfer

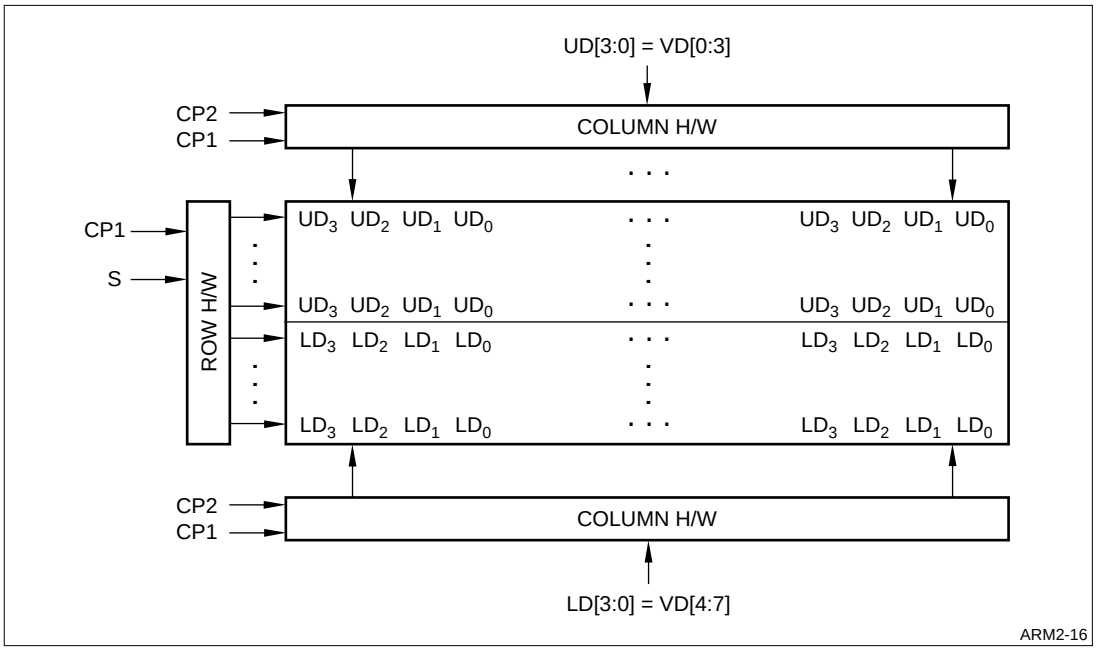


Figure 10-4. Dual Scan Mode (Always 4-Bit Transfer)

CP2

CP2 is the Shift/Transfer clock for the display data VD[7:0]. The contents of VD[7:0] are shifted/transferred into the Column hardware on the falling edge of CP2.

CP1

CP1 is the Line Pulse/Horizontal Sync signal. This signal will be activated when all the display data associated with a line are transferred to the LCD Panel column hardware. The falling edge of CP1 will signal the LCD Panel to display the transferred line.

S

This is the Frame Clock/Vertical Sync signal. It signals to the LCD panel the start of a new frame (a new screen update). It is active High and the LCD panel samples it on the falling edge of CP1.

MCLK

This signal provides AC Modulation for the LCD drivers to ensure no DC build up which will cause chemical breakdown of the LCD fluid and ruin the panel. MCLK is a square wave signal that will allow the LCD to be driven for equal periods of time at positive and negative polarity. The frequency of the MCLK is programmable via LCD_MCLKW Register.

Today's panels generally will toggle this signal in the range of 13 to 30 line times (CP1). Most SHARP LCD panels will generate this signal internally based on CP1 and would not need an external MCLK.

LCDCNTL

This signal reflects the value of the LCDC bit in LCD_MODE register (bit 1) as follows:

LCD_MODE(1) = 0 -----> LCDCNTL = 0

LCD_MODE(1) = 1 -----> LCDCNTL = 1

This is a general purpose control signal. Some of the possible usage of this signal would be:

1. Power ON/OFF sequence of LCD panels
2. Turn on/off the Backlight of LCD panels to save power

REGISTER MAP

The base address for LCD register(s) is 0xFFFF1400. The offset, initial value, access and number of bits for each register are as follows.

Table 10-3.
LCD Controller Register Map

REGISTER NAME	ADDRESS OFFSET [7:0]	ACCESS	RESET VALUE
LCD_MODE[7:0]	00	W	0x00
LCD_BC[7:0]	04	W	0x00
LCD_CP1W[7:0]	08	W	0x00
LCD_DUTY[7:0]	0C	W	0x00
LCD_DUTY[9:8]	10	W	0x00
LCD_SADR1[7:0]	14	W	0x00
LCD_SADR1[15:8]	18	W	0x00
LCD_SADR1[23:16]	1C	W	0x00
LCD_SADR1[31:24]	20	W	0x00
LCD_SADR2[7:0]	24	W	0x00
LCD_SADR2[15:8]	28	W	0x00
LCD_SADR2[23:16]	2C	W	0x00
LCD_SADR2[31:24]	30	W	0x00
LCD_VLC1[7:0]	34	W	0x00
LCD_VLC1[9:8]	38	W	0x00
LCD_VDLT[7:0]	3C	W	0x00
LCD_GRAY1[7:0]	40	W	0x00
LCD_GRAY2[7:0]	44	W	0x00
LCD_CLKDIV[7:0]	48	W	0x00
LCD_MCLKW [7:0]	4C	W	0x00
LCD_MCLKW [9:8]	50	W	0x00

The base address for LCD Bit Control register is 0xFFFFA414. The offset, initial value, access and number of bits for the register is as follows

REGISTER NAME	ADDRESS OFFSET [7:0]	ACCESS	RESET VALUE
LCD_BITCTL	00	R/W	0x00

DESCRIPTION OF REGISTERS

Mode Register (LCD_MODE)

7	6	5	4	3	2	1	0
DISP	REV	SCAN	OR	GRAY	XSIZE	LCDC	LCDA

LCD Register LCD_MODE controls the operational modes of the LCD controller. The individual bits are defined as follows:

Bit 7 – DISP

DISP turns the display ON or OFF, according to Table 10-4.

Table 10-4.
LCD_MODE, DISP Bit

DISP	REV	VD[7:0]
0	0	0x00
1	0	DATA
0	1	0xFF
1	1	$\overline{\text{DATA}}$

NOTE: Data is the frame buffer data

The LCD controller keeps fetching data from the frame buffer regardless of the state of the DISP bit.

Bit 6 – REV

REV selects between Normal and Reverse display modes, according to Table 10-5.

Table 10-5.
LCD_MODE, REV Bit

DISP	REV	VD[7:0]
0	0	0x00
1	0	DATA
0	1	0xFF
1	1	$\overline{\text{DATA}}$

NOTE: Data is the frame buffer data

Bit 5 – SCAN

SCAN selects between Single and Dual scan modes, according to Table 10-6.

Table 10-6.
LCD_MODE, SCAN Bit

PANEL	MODE
0	Single Scan
1	Dual Scan

Bit 4 – OR

OR is only valid when the LCD controller is in single scan mode.

1. Division
2. Logically ORing the data from two screens

This bit is *valid only* in *Single Scan mode*. Table 10-7 describes the function of the OR bit.

Table 10-7.
LCD_MODE, OR Bit

OR	MODE
0	Division: The LCD Panel is divided into two screens. The upper screen has LCD_SADR1 as the starting address of its frame buffer. The lower screen has LCD_SADR2 as the starting address of its frame buffer. The number of lines on each screen is determined by the value of LCD_VLC1 and LCD_DUTY. In order to force a single screen display, set LCD_VLC1 ≥ LCD_DUTY.
1	OR: The LCD Panel has one screen that displays the bit-by-bit OR function of two frame buffers. The first frame buffer has LCD_SADR1 as the starting address. The second frame buffer has LCD_SADR2 as the starting address. In Binary mode, the data from frame buffer 1 is bit-by-bit ORed with the data from frame buffer 2. In Gray mode, the converted data from frame buffer 1 (after applying LCD_GRAY1) and the converted data from frame buffer 2 (after applying LCD_GRAY2) is bit-by-bit ORed.

Bit 3 – GRAY

The Gray bit selects between binary (two shades) and gray (4 shades) display modes, according to Table 10-8.

Table 10-8.
LCD_MODE, GRAY Bit

GRAY	MODE
0	Binary Display Mode: One bit of data in the frame buffer is used to drive each pixel on the display. There are two shades (ON/OFF).
1	Gray Display Mode: Two bits of data in the frame buffer are used to drive each pixel on the display. There are 4 shades of gray (ON, OFF, GRAY1, GRAY2)

Bit 2 - XSIZE

The XSIZE selects either 4-bit or 8-bit data transfers (per CP2 clock) to the LCD display, according to Table 10-9.

Table 10-9.
LCD_MODE, XSIZE

XSIZE	DATA TRANSFER PER CP2 CLOCK
0	4-Bit
1	8-Bit

8-bit data transfer is only used in binary/single scan mode. Table 10-10 summarizes the transfer size relationship to both the SCAN bit and GRAY bit in LCD_MODE register

Table 10-10.
Transfer Size

SCAN	GRAY	TRANSFER SIZE
0	0	4 or 8 bits
0	1	4 bits
1	0	4 bits
1	1	4 bits

Bit 1 - LCDC

The LCDC bit controls the state of the LCDCNTL output signal. The LCDCNTL signal is a general purpose control signal as defined earlier. The function of the LCDC bit is described in Table 10-11.

Table 10-11.
LCD_MODE, LCDC

LCDC	MODE
0	LCDCNTL = 0
1	LCDCNTL = 1

Bit 0 – LCDA

The LCDA bit controls the activation of the LCD controller. Upon reset, this bit is set to '0' disabling the LCD controller internal state machines and thus the controller's output signals (CP1, CP2, S, VD, MCLK, LCDCNTL). LCD_MODE register should be the last LCD register to be programmed with LCDA bit set to '1'. This will activate the LCD controller and thus the controller's output signals (CP1, CP2, S, VD, MCLK, LCDCNTL). The LCD controller can be reprogrammed while it is active. LCD_CLKDIV can only be programmed if LCDA = 0.

Table 10-12.
LCD_MODE, LCDA

LCDA	LCD CONTROLLER
0	Not Active
1	Active

The user should pay close attention to the power on/off sequence of the LCD panel to avoid ruining the LCD panel. Upon reset, and if the LCD panel already has power applied, the LCD controller should be programmed and activated as soon as possible.

When $\overline{WAIT}$ input on the 790A are asserted active, the LCD controller will stall when it runs out of data. This will freeze the LCD outputs (clock and data) which would result in DC buildup.

LCD_MODE Register Summary

Tables 10-13 and 10-14 summarize the LCD_MODE bits and the eight display modes.

Table 10-13.
LCD_MODE Register Summary

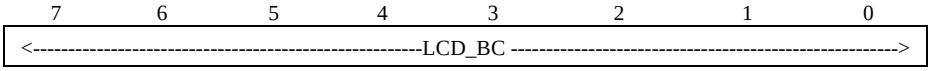
BIT	SYMBOL	VALUE	MODE
7	DISP	0 1	Display OFF Display ON
6	REV	0 1	Normal Display Reverse Display
5	SCAN	0 1	Single Scan Dual Scan
4	OR	0 1	Division OR
3	GRAY	0 1	Binary Display Gray Display
2	XSIZE	0 1	4-Bits 8-Bits
1	LCDC	0 1	LCDCNTL = 0 LCDCNTL = 1
0	LCDA	0 1	Not Active Active

Table 10-14.
LCD Controller Display Modes

DISPLAY MODE	D7	D6	D5	D4	D3	D2	D1	D0			
	DISP	REV	SCAN	OR	GRAY	XSIZE	LCDC	LCDA			
1a (4-Bit)	X	X	0	0	0	0	X	1	Binary	Division	Single Scan
1b (8-Bit)	X	X	0	0	0	1	X	1			
2	X	X	0	0	1	X	X	1	Gray		
3a (4-Bit)	X	X	0	1	0	0	X	1	Binary	OR	
3b (8-Bit)	X	X	0	1	0	1	X	1			
4	X	X	0	1	1	X	X	1	Gray		
5	X	X	1	X	0	X	X	1	Binary		Dual Scan
6	X	X	1	X	1	X	X	1	Gray		

NOTE: X = Don't Care

Line Display Byte Count Register (LCD_BC)



LCD_BC register controls the number of display bytes per line in the frame buffer. In binary display mode (LCD_MODE(3)=0), a byte in the frame buffer reflects 8 LCD pixels (1-bit/pixel). In the gray display mode, a byte in the frame buffer reflects 4 LCD pixels (2-bits/pixel). So for a given LCD panel of width W pixels, LCD_BC is programmed as W ÷ 8 in binary display mode or 2 × W ÷ 8 in gray display mode.

The following table shows the relationship between LCD_BC and the number of display pixels. Note that the binary value of the register is the number of bytes less one.

Table 10-15.
Relation Between LCD_BC and the Number of Display Pixels

LCD_BC[7:0] (BINARY)								NO. OF BYTES (DECIMAL)	DISPLAY PIXELS	
7	6	5	4	3	2	1	0		BINARY MODE	GRAY SCALE
0	0	0	0	0	0	0	0	1*	8	N/A
0	0	0	0	0	0	0	1	2	16	8
0	0	0	0	0	0	1	0	3*	24	N/A
0	0	0	0	0	0	1	1	4	32	16
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
0	1	0	0	1	1	1	1	80	640	320
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
1	0	0	1	1	1	1	1	160	1280	640
.	.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	.	
1	1	1	1	1	1	0	1	254	2032	1016
1	1	1	1	1	1	1	0	255*	2040	N/A
1	1	1	1	1	1	1	1	256	2048	1024

NOTE:

1. *LCD Panels that are not a multiple of 16 in width (8, 24, ..., 328, ..., 2040). See next paragraph and example.
2. N/A = Not Available

The LCD controller is designed to support LCD panels' widths that are a multiple of 16 (16, 32, 48, ..., 640, ..., 2048) and the majority of standard panels are a multiple of 16. For LCD panels widths that are a multiple of 8 but not 16 (8, 24, ..., 328, ... 2040), the LCD controller will still function properly in binary mode only if each line in the frame buffer is padded with an extra byte of any value. The padded byte will not be displayed but will guarantee proper display of the original image. LCD_BC should still be calculated using the original panel width prior to the padding.

Example – 320 pixel wide display:

Binary Display Mode:

LCD_BC = (320 pixels/line) × (1 bit/pixel) × (1 Byte)/(8 bits) = 40 bytes/line

Gray Display Mode:

LCD_BC = (320 pixels/line) × (2 bits/pixel) × (1 Byte)/(8 bits) = 80 bytes/line

Example - 24 pixel wide display

Binary Display Mode:

LCD_BC = (24 pixels/line) × (1 bit/pixel) × (1 Byte)/(8 bits) = 3 bytes/ line

In order to display the frame buffer properly, the frame buffer need to be modified as follows:

Table 10-16.
Padded Frame Buffer Example

ORIGINAL FRAME BUFFER	MODIFIED FRAME BUFFER
Line 0, Byte 0	Line 0, Byte 0
Line 0, Byte 1	Line 0, Byte 1
Line 0, Byte 2	Line 0, Byte 2
Line 1, Byte 0	Line 0, Padded Byte
Line 1, Byte 1	Line 1, Byte 0
Line 1, Byte 2	Line 1, Byte 1
Line 2, Byte 0	Line 1, Byte 2
Line 2, Byte 1	Line 1, Padded Byte
Line 2, Byte 2	Line 2, Byte 0

Line Pulse Width Register (LCD_CP1W)

7	6	5	4	3	2	1	0
<----------LCD_CP1W---------->							

LCD_CP1W controls the width of the line pulse high time, CP1, as a function of CP2. This allows the frame frequency to be adjusted to drive various LCD panels. Refer to the Basic Timing Section for more detail.

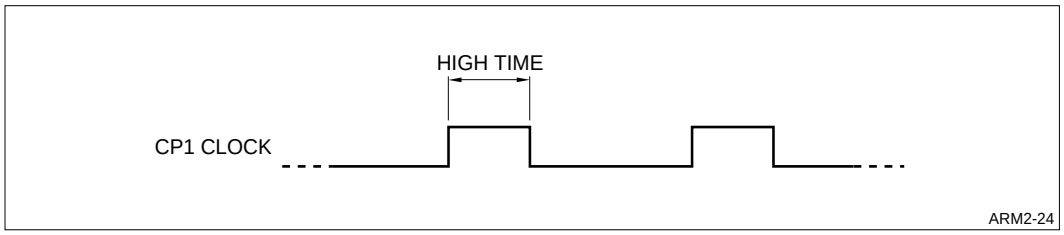


Figure 10-5. CP1 Clock Pulse High Time

Table 10-17.
Line Pulse Width

LCD_CP1W (BINARY)								LCD_CP1W (DECIMAL)
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
0	0	0	0	0	0	0	0	illegal
0	0	0	0	0	0	0	1	2*
0	0	0	0	0	0	1	0	4**
0	0	0	0	0	0	1	1	6
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
1	1	1	1	1	1	0	1	506
1	1	1	1	1	1	1	0	508
1	1	1	1	1	1	1	1	510

NOTES:
*Illegal in modes 1a, 3a and 5.
** Illegal in Modes 3a and 5.

Duty Cycle Registers (LCD_DUTY)

7	6	5	4	3	2	1	0
<----------LCD_DUTY[7:0] ---------->							
// <-LCD_DUTY[9:8]->							

LCD_DUTY registers define the total number of CP1 pulses per frame which may be equal to or greater than the duty cycle, depending on the LCD panel. In single scan mode, LCD panels typically have the duty cycle equal to the total number of vertical display lines. For example, a SHARP LM32P10 has a resolution of 320 (H) × 240 (V) and a duty cycle of 1/240. In dual scan mode, LCD panels are divided in half into upper and lower screens that are driven in parallel. This means that the duty cycle will be typically 1/2 the total number of vertical display lines. For example, a SHARP LM64K101 has a resolution of 640 (H) × 480 (V) and a duty cycle of 1/240, but requires 241 CP1 pulses (LCD_DUTY = 0x0F0) to operate properly. On the other hand, EPSON EG9005F-LS has a resolution of 640 (H) × 480 (V) and a duty cycle of 1/242. The LCD controller will request data from external memory for every line programmed in the LCD_DUTY register.

Table 10-18.
Relation Between (LCD_DUTY) and Number of CP1 Pulses Per Frame

LCD_DUTY [9:0]* (BINARY)										# OF CP1 PULSES PER FRAME (DECIMAL)
9	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
0	0	0	0	0	0	0	0	0	0	Illegal
0	0	0	0	0	0	0	0	0	1	Illegal
0	0	0	0	0	0	0	0	1	0	3
0	0	0	0	0	0	0	0	1	1	4
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
0	0	0	1	1	0	0	0	1	0	100
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
0	0	1	1	0	0	0	1	1	1	200
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
0	0	1	1	1	0	1	1	1	1	240
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
0	1	1	1	1	1	1	1	1	1	512
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
1	1	1	1	1	1	1	1	0	1	1022
1	1	1	1	1	1	1	1	1	0	1023
1	1	1	1	1	1	1	1	1	1	1024

NOTE: LCD_DUTY uses two registers to form a 10-bit value.

Examples :

SHARP LM32P07 (320 × 240, Single Scan, Duty Cycle = 1/240) will have LCD_DUTY [9:0] = 0x0EF

SHARP LM6K101 (640 x 480, Dual Scan, Duty Cycle = 1/240) will have LCD_DUTY [9:0] = 0x0F0

EPSON EG9005F-LS (640 x 480, Duty Cycle = 1/242) will have LCD_DUTY [9:0] = 0x0F1

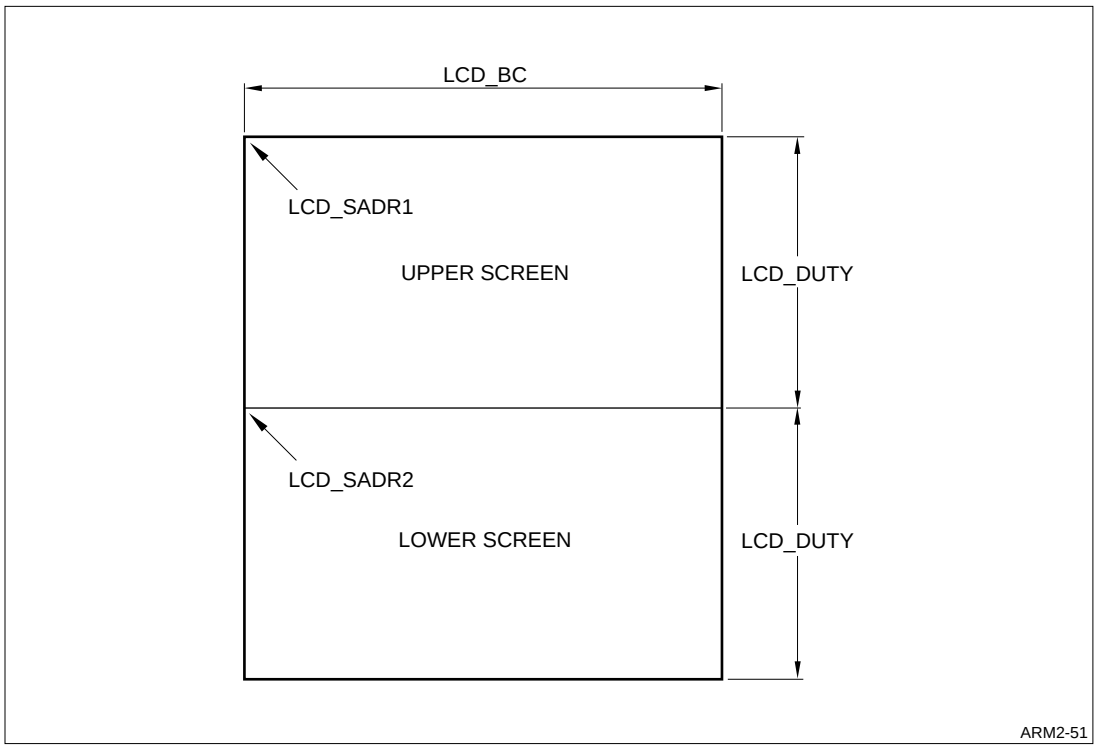


Figure 10-6. LCD Panel in Dual Mode

Screen #1 Frame Buffer Start Address Registers (LCD_SADR1[31:0])

7	6	5	4	3	2	1	0
<----------LCD_SADR1[7:0] ---------->							
<----------LCD_SADR1[15:8] ---------->							
<---------- LCD_SADR1[23:16] ---------->							
<----------LCD_SADR1[31:24] ---------->							

LCD_SADR1 registers define the 32-bit start address of the frame buffer for screen #1. The setting can range from 0x00000000 to 0xFFFFFFFE. The starting address is half-word aligned, so the LSB bit must always be '0'. The definition of Screen #1 depends on the display mode as shown in Table 10-19.

The frame buffer should always reside in external memory. LCD requests are treated by the bus controller as system level requests. The frame buffer segment should have at least a system read privilege and should not be cacheable. The frame buffer should be large enough to hold the image and the extra lines in the case where the duty cycle is larger than the number of vertical lines.

Table 10-19.
Relationship between Screen #1 and Display Modes

DISPLAY MODE	SCREEN #1
1a, 1b, 2	Upper Screen Frame Buffer
3a, 3b, 4	First Frame Buffer to be ORed
5, 6	Upper Screen Frame Buffer

Screen #2 Frame Buffer Start Address Registers (LCD_SADR2)

7	6	5	4	3	2	1	0
<---------- LCD_SADR2[7:0] ---------->							
<---------- LCD_SADR2[15:8] ---------->							
<---------- LCD_SADR2[23:16] ---------->							
<---------- LCD_SADR2[31:24] ---------->							

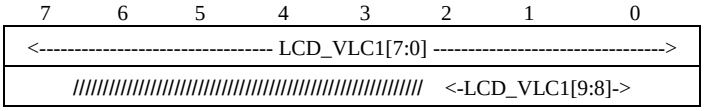
LCD_SADR2 registers define the 32-bit start address of the frame buffer for screen #2. The setting can range from 0x00000000 to 0xFFFFFFFE. The starting address is half-word aligned, so the LSB bit must always be '0'. The definition of Screen #2 depends on the display mode as shown in Table 10-20.

The frame buffer should always reside in external memory. LCD requests are treated by the bus controller as system level requests. The frame buffer segment should have a least a system read privilege and should not be cacheable. The frame buffer should be large enough to hold the image and the extra lines in the case where the duty cycle is larger than the number of vertical lines.

Table 10-20.
Relationship Between Screen #2 and Display Modes

DISPLAY MODE	SCREEN #2
1a, 1b, 2	Lower Screen Frame Buffer
3a, 3b, 4	Second Frame Buffer to be ORed
5, 6	Lower Screen Frame Buffer

Screen #1 Vertical Line Count Registers (LCD_VLC1)



// <-LCD_VLC1[9:8]->

LCD_VLC1 registers define the number of vertical lines on screen #1. They are used only in the single scan mode with the panel divided onto two screens (display modes 1a, 1b, and 2). LCD_VLC1 is the number of lines on the upper screen defined by screen #1 display start address, LCD_SADR1. The number of vertical lines on the lower screen, screen #2, is the difference between LCD_DUTY and LCD_VLC1. Figure 10-7 shows an example of a panel in division mode with two screens. In order to force a single screen, set LCD_VLC1 $\geq$ LCD_DUTY.

Table 10-21.
Relationship Between (LCD_VLC1) and Screen No. 1 Lines

LCD_VLC1[9:0]* (BINARY)										SCREEN #1 VERTICAL LINES (DECIMAL)
9	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1	2
0	0	0	0	0	0	0	0	1	0	3
0	0	0	0	0	0	0	0	1	1	4
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
1	1	1	1	1	1	1	1	0	1	1022
1	1	1	1	1	1	1	1	1	0	1023
1	1	1	1	1	1	1	1	1	1	1024

NOTE: LCD_VLC1 uses two registers to form a 10-bit Value.

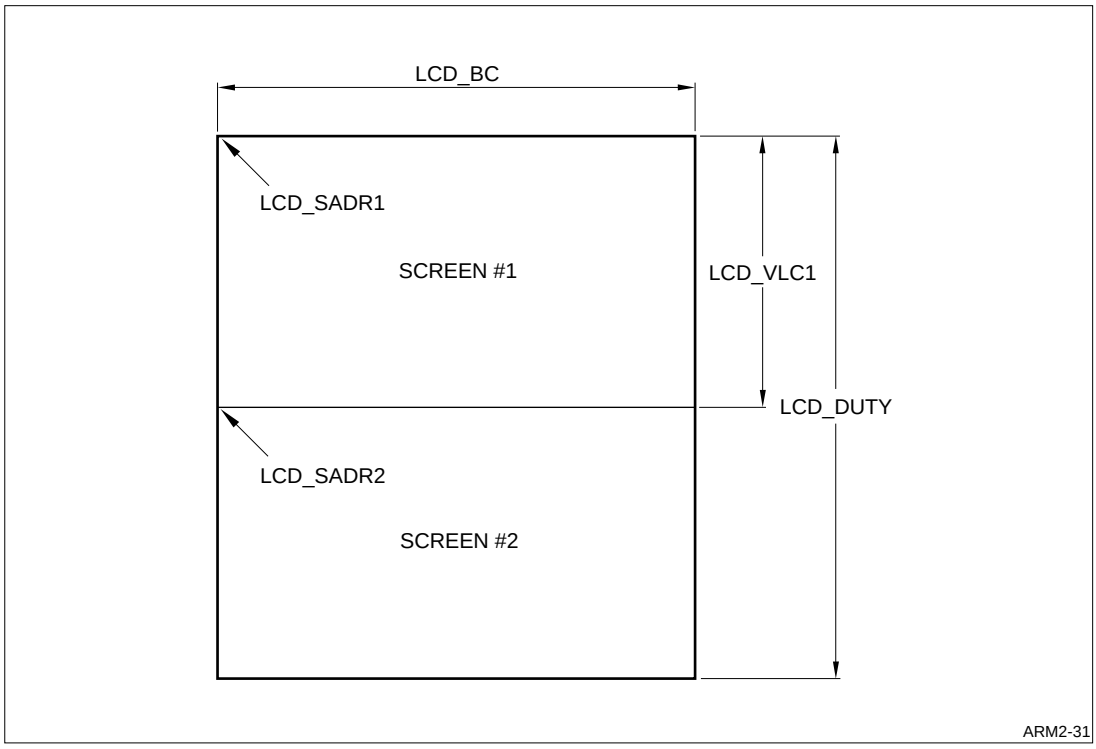


Figure 10-7. LCD Panel in Division Mode

Virtual Display Delta Register (LCD_VDLT)

7	6	5	4	3	2	1	0
<---------- LCD_VDLT[7:0] ---------->							

LCD_VDLT register is the difference between the address of the last halfword displayed on the previous LCD line (A_{old}) and the address of the first halfword to be displayed on the new LCD line (A_{new}).

$LCD\_VDLT = A_{new} - A_{old}$

Total number of bytes/frame buffer line = LCD_BC+LCD_VDLT-2

The address of the first halfword to be displayed on each new LCD line is computed by adding LCD_VLDT value to the address of the last halfword displayed on the previous LCD line. By changing the starting address of the frame buffer, a scrolling function can be implemented.

A virtual screen is when an image stored in memory is larger than the LCD panel size. LCD_VDLT allows the virtual image to be displayed. A portion of the image is actually displayed and the rest of the image can be displayed by scrolling horizontally. The width of the virtual display panel is 510 bytes (LCD_BC+LCD_VDLT-2). So the controller can support 4080 pixels in binary mode and 2040 pixels in gray mode.

Table 10-22.
Relationship Between LCD_VDLT and Address Difference

LCD_VDLT								ADDRESS DIFFERENCE (DECIMAL)
7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	X	2
0	0	0	0	0	0	1	X	4
0	0	0	0	0	1	0	X	6
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
1	1	1	1	1	0	1	X	252
1	1	1	1	1	1	0	X	254
1	1	1	1	1	1	1	X	256

NOTE: X = Don't Care

Example 1 :

Figure 10-8 shows an image displayed on the LCD panel before scrolling. Figure 10-9 shows the image displayed after scrolling.

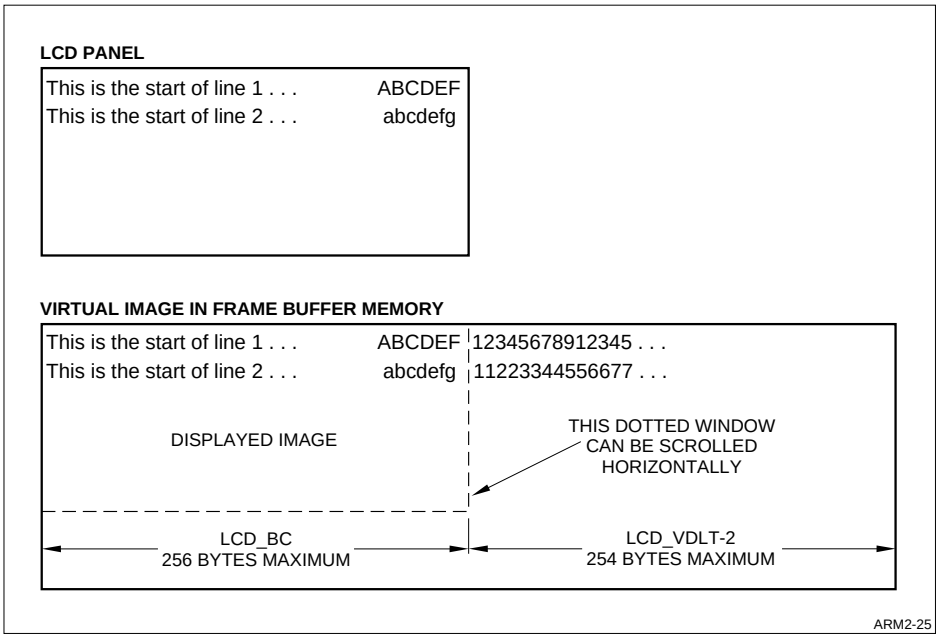


Figure 10-8. LCD Panel Before Scrolling

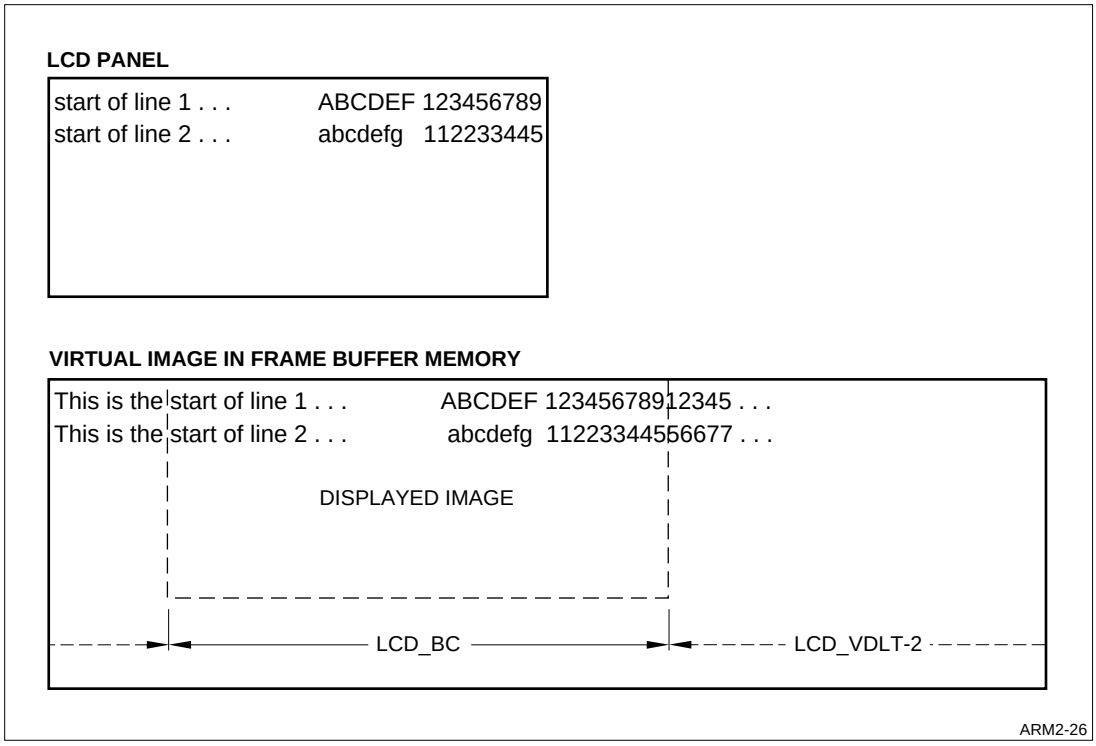


Figure 10-9. LCD Panel after Scrolling

Example 2 :

If a 320x4 LCD Panel is used, Binary Mode, and the virtual display is 120 bytes wide then:

LCD_BC = 320/8 = 40 bytes (0x27)

LCD_VDLT = 120 - 40 + 2 = 82 bytes (0x50)

Gray Shade Registers (LCD_GRAY1, LCD_GRAY2)

7	6	5	4	3	2	1	0
<----------LCD_GRAY1[7:0] ---------->							
<----------LCD_GRAY2[7:0] ---------->							

LCD_GRAY1 and LCD_GRAY2 registers control the gray shades on the LCD panel when the LCD controller is running in gray display mode (modes 2,4,6). In this mode, every 2 bits of a byte of data in the frame buffer form a pair (x,y) as shown in Table 10-23 (refer to LCD_BITCTL section for definition of Format #0 and Format #1):

Table 10-23.
Relationship between Frame Buffer Byte, (x,y) Pair and LCD Bit Format

Frame Buffer Byte	b7	b6	b5	b4	b3	b2	b1	b0
Paired Bits	(x,y)		(x,y)		(x,y)		(x,y)	

Format #0

Frame Buffer Byte	b7	b6	b5	b4	b3	b2	b1	b0
Paired Bits	(y,x)		(y,x)		(y,x)		(y,x)	

Format #1

Each pair (x, y) corresponds to a single pixel to be displayed in 4 shades according to Table 10-24.

Table 10-24.
Four Gray Shades

X	Y	LCD DISPLAY
0	0	Display OFF
0	1	Display Gray Shade based on LCD_GRAY1
1	0	Display Gray Shade based on LCD_GRAY2
1	1	Display ON

From the above table, (x,y) = (0,1) will select LCD_GRAY1 and (x,y) = (1,0) will select LCD_GRAY2. As shown below, LCD_GRAY1 and LCD_GRAY2 are both loaded into a shift register that is clocked by the frame clock, S. The outputs of the shift registers are fed into selectors that has 4 inputs (0, 1, LCD_GRAY1, LCD_GRAY2). The outputs of the selectors are connected to the output display data bits, VD[7:0]. Each pair of bits from the frame buffer will select one of four inputs to connect to VD[7:0]. Every frame clock, a new bit of LCD_GRAY1 and LCD_GRAY2 is shifted out and provided as an input to the selector.

NOTE: Certain combination of gray patterns and MCLK frequencies may cause pixels to always switch ‘on’ during the same polarity phase of MCLK which would cause DC buildup.

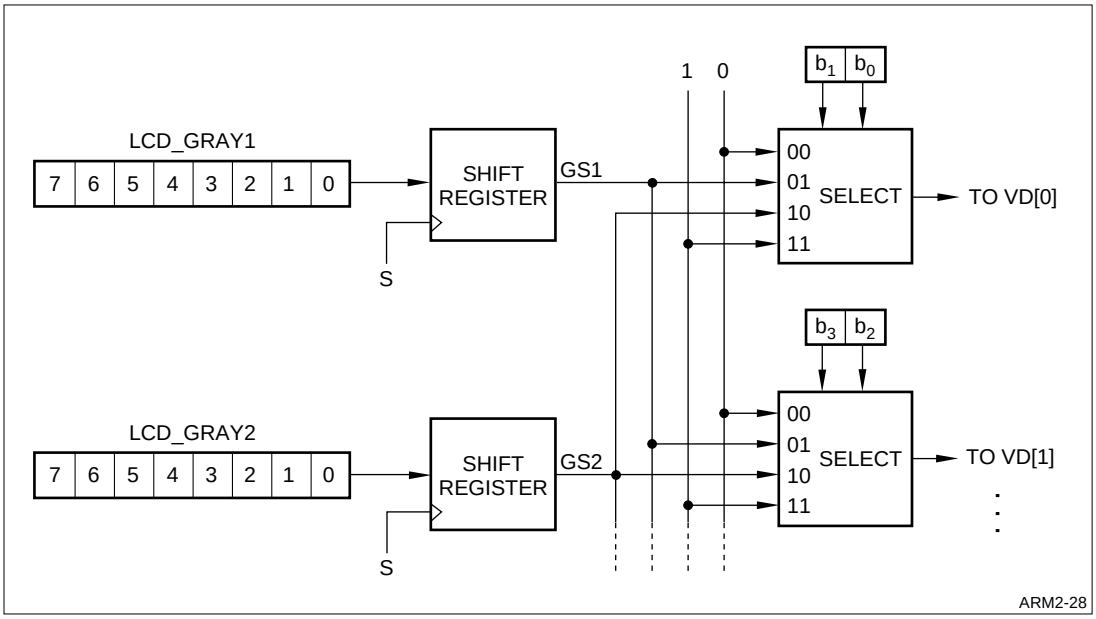


Figure 10-10. Format 0 Gray Shade Logic

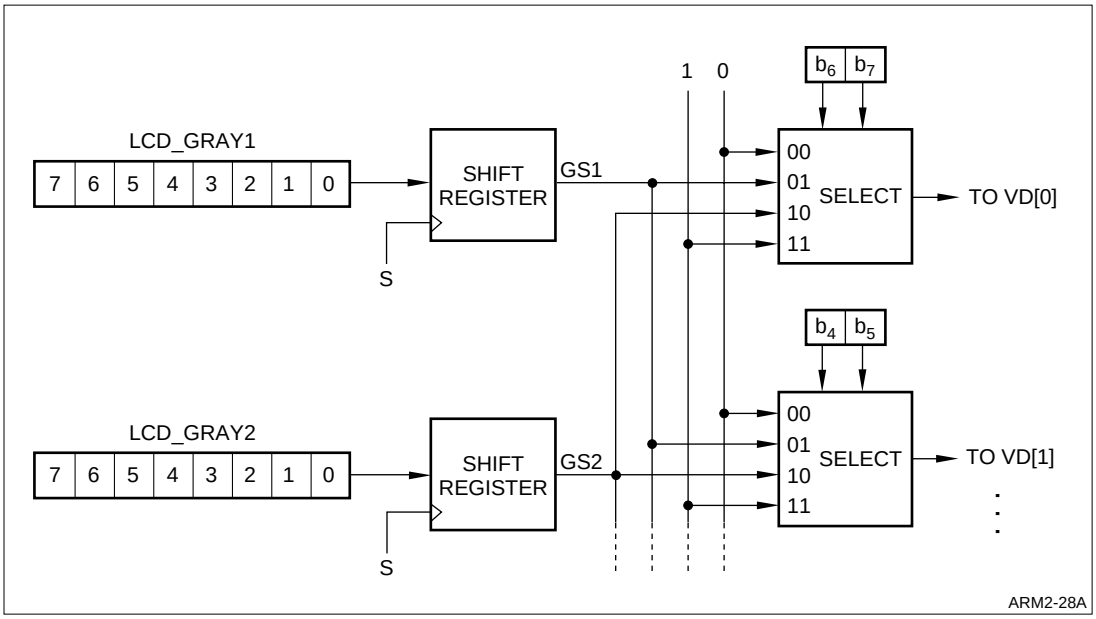


Figure 10-11. Format 1 Gray Shade Logic

In Format #1, if it is desirable to use (x,y) = (b7,b6) for gray mode, the contents of LCD_GRAY1 and LCD_GRAY2 registers can be swapped.

Example 1:

To better understand the operation, let's assume that pixel 1 has (x,y) = (0,1), and LCD_GRAY1 is '01101101'. That means that LCD_GRAY1 pattern will be chosen every time pixel 1 is displayed (once every frame). On the first frame clock, VD[0] will get LCD_GRAY1[0], '1'. On the second frame clock, VD[0] will get LCD_GRAY1[1], '0'. On the third frame clock, VD[0] will get LCD_GRAY1[2], '1' and so on. On the ninth frame clock, the LCD_GRAY1 sequence repeats itself.

Table 10-25.
Example of Pixel 1 State in Gray Scale Mode

FRAME CLOCK	VD[0]	PIXEL 1 ON LCD PANEL
1	1	On
2	0	Off
3	1	On
4	1	On
5	0	Off
6	1	On
7	1	On
8	0	Off
9	1	On
10	0	Off
11	1	On
:	:	:

A gray shade will develop as the pixel state (value) changes every frame clock. By the 8th frame, the pixel will develop a gray shade. This is true because the liquid crystal in the display is slow to react to the changes in the pixel state from frame to frame.

The gray shade is a function of frame rate, LCD_GRAY1 and LCD_GRAY2 patterns and the LCD panel characteristics. The above method is referred to as Frame Rate Duty Cycle (FRDC). Certain combination of gray patterns and MCLK frequencies can cause pixels to only turn ON when MCLK clock is in one polarity phase, causing DC buildup.

Clock Frequency Divider (LCD_CLKDIV)

LCD_CLKDIV register internally divides the input clock to the LCD controller, LCD_in_CLK, to produce the reference clock, lcdclk. The reference clock, lcdclk, is used to generate all the timing signals for the LCD panel (S,CP1,CP2,MCLK).

$$t_{lcdclk} = t_{LCD\_in\_CLK} \times LCD\_CLKDIV$$

LCD_in_CLK is a gated XCLK clock generated by the Clock and Power Management Unit.

Table 10-26.
Relationship Between LCD_CLKDIV and Divisor Value

LCD_CLKDIV								DIVISOR (DECIMAL)
7	**6**	**5**	**4**	**3**	**2**	**1**	**0***	
0	0	0	0	0	0	0	0	2
0	0	0	0	0	0	1	0	4
0	0	0	0	0	1	0	0	6
.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	
1	1	1	1	1	0	1	0	252
1	1	1	1	1	1	0	0	254
1	1	1	1	1	1	1	0	256

NOTE: LCD_CLKDIV (0) must always be 0

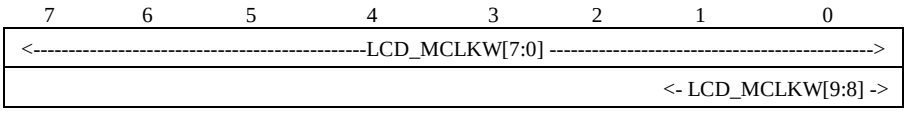
Example 2 :

LCD_CLKDIV = '00000010'

$t_{LCD\_in\_CLK}$ = 50 ns (20 MHz System Clock)

$t_{lcdclk} = 50 \times 4 = 200$ ns (5 MHz Reference Clock)

MCLK Width (LCD_MCLKW)



<- LCD_MCLKW[9:8] ->

LCD_MCLKW register is used to program the MCLK width (MCLKW) relative to CP1 clock. The MCLK converts the LCD driver voltage to an AC voltage by inverting it periodically. This will keep the LCD panel from being driven at a DC level which will cause chemical breakdown of the LCD fluid and ruin the panel (DC Buildup). Since the frequency of inversion varies from one LCD panel to another, this register allows the user to program the MCLK width. MCLK changes phases on a falling edge of CP1.

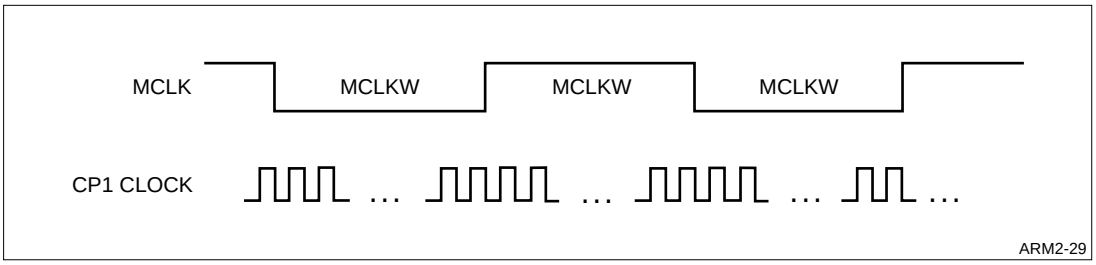


Figure 10-12. MCLK and CP1 Relationship

Table 10-27.
Relationship Between LCD_MCLKW and CP1

LCD_MCLKW* [9:0]										WIDTH OF MCLK IN CP1 CLOCKS (MCLKW) (DECIMAL)
9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1	2
0	0	0	0	0	0	0	0	1	0	3
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
1	1	1	1	1	1	1	1	0	1	1022
1	1	1	1	1	1	1	1	1	0	1023
1	1	1	1	1	1	1	1	1	1	1024

NOTE: that LCD_MCLKW uses two registers to form a 10-bit value.

So if LCD_MCLKW is set to '000001100', the MCLK will invert polarity every 13 CP1 clocks. Some LCD panels, e.g. SHARP panels, produce the MCLK internally (based on CP1) and do not require this signal. Care should be taken to avoid DC Buildup. DC buildup can be avoided by:

1. Having the MCLK width be a larger multiple of CP1 (greater than 12).
2. Having MCLK change polarity on different CP1 pulses from one frame to another.
3. Avoid having pixels switch ‘on’ on the same polarity phase of MCLK every frame.
4. Having a 50% (± tolerance) duty cycle on MCLK (internally or externally generated).

:

Consult with the panel specification/manufacturer on the best value to program LCD_MCLKW.

LCD Bit Control (LCD_BITCTL)

7	6	5	4	3	2	1	0
////	////	////	////	////	////	////	LBC

LCD_BITCTL allows the frame buffer to be displayed in two different formats. In binary mode, Format #0 puts bit(0) of any frame buffer byte on VD(0), bit(1) on VD(1) and so on. Format #1 puts bit(7) of any frame buffer byte on VD(0), bit(6) on VD(1) and so on. Both formats displays byte(0) first, followed by byte(1) and so on.

LCD_BITCTL register when set to 1, swaps bit(0) with bit(7), bit(1) with bit(6), bit(2) with bit(5) and bit(3) with bit(4) of each byte read from the frame buffer before it is delivered to the LCD controller for processing. LCD_BITCTL register when set to 0, will pass the frame buffer data with no modification to the LCD controller for processing.

Table 10-28.
LCD_BITCTL, LBC Bit

LBC	FUNCTION
0	0: Display Frame Buffer using Format #0
1	1: Display Frame Buffer using Format #1

The following example will illustrate the differences between Format #0 and Format #1:

Example

Assumptions:

Assuming that the frame buffer to be displayed is stored in main memory as follows:

Table 10-29.
Frame Buffers in Main Memory

BYTE ADDRESS	DISPLAY DATA
:	:
10000003	$D_7D_6D_5D_4D_3D_2D_1D$
10000002	$C_7C_6C_5C_4C_3C_2C_1C$
10000001	$B_7B_6B_5B_4B_3B_2B_1B_0$
10000000*	$A_7A_6A_5A_4A_3A_2A_1A_0$

BYTE ADDRESS	DISPLAY DATA
:	:
20000003	$d_7d_6d_5d_4d_3d_2d_1d_0$
20000002	$c_7c_6c_5c_4c_3c_2c_1c_0$
20000001	$b_7b_6b_5b_4b_3b_2b_1b_0$
20000000**	$a_7a_6a_5a_4a_3a_2a_1a_0$

NOTE:
*Frame Buffer 1 Starting Address
** Frame Buffer 2 Starting Address

Results:

Selecting **Format #0** will put the frame buffer data bits on the VD bus as follows:

LCD MODE	TRANS-FER	VD(7)	VD(6)	VD(5)	VD(4)	VD(3)	VD(2)	VD(1)	VD(0)
SS, 4-bit	1st	0	0	0	0	A_3	A_2	A_1	A_0
	2nd	0	0	0	0	A_7	A_6	A_5	A_4
SS, 8-bit	1st	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	2nd	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
SS, Gray	1st	0	0	0	0	$G(A_7A_6)$	$G(A_5A_4)$	$G(A_3A_2)$	$G(A_1A_0)$
	2nd	0	0	0	0	$G(B_7B_6)$	$G(B_5B_4)$	$G(B_3B_2)$	$G(B_1B_0)$
DS, Binary	1st	a_3	a_2	a_1	a_0	A_3	A_2	A_1	A_0
	2nd	a_7	a_6	a_5	a_4	A_7	A_6	A_5	A_4
DS, Gray	1st	$G(a_7a_6)$	$G(a_5a_4)$	$G(a_3a_2)$	$G(a_1a_0)$	$G(A_7A_6)$	$G(A_5A_4)$	$G(A_3A_2)$	$G(A_1A_0)$
	2nd	$G(b_7b_6)$	$G(b_5b_4)$	$G(b_3b_2)$	$G(b_1b_0)$	$G(B_7B_6)$	$G(B_5B_4)$	$G(B_3B_2)$	$G(B_1B_0)$

NOTE:

G: indicates that gray shade is applied to the pair
SS: Single Scan & DS: Double Scan

Advantage: Simple and fast data shifts and bit index calculations.
Disadvantage: Displayed byte is not in the same order as the stored byte

Selecting **Format #1** will put the frame buffer data bits on the VD bus as follows:

LCD MODE	TRANS-FER	VD(7)	VD(6)	VD(5)	VD(4)	VD(3)	VD(2)	VD(1)	VD(0)
SS, 4-bit	1st	0	0	0	0	A_4	A_5	A_6	A_7
	2nd	0	0	0	0	A_0	A_1	A_2	A_3
SS, 8-bit	1st	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7
	2nd	B_0	B_1	B_2	B_3	B_4	B_5	B_6	B_7
SS, Gray[1]	1st	0	0	0	0	$G(A_0A_1)$	$G(A_2A_3)$	$G(A_4A_5)$	$G(A_6A_7)$
	2nd	0	0	0	0	$G(B_2B_3)$	$G(B_2B_3)$	$G(B_4B_5)$	$G(B_6B_7)$
DS, Binary	1st	a_4	a_5	a_6	a_7	A_5	A_5	A_6	A_7
	2nd	a_0	a_1	a_2	a_3	A_1	A_1	A_2	A_3
DS, Gray[1]	1st	$G(a_0a_1)$	$G(a_2a_3)$	$G(a_4a_5)$	$G(a_6a_7)$	$G(A_2A_3)$	$G(A_2A_3)$	$G(A_4A_5)$	$G(A_6A_7)$
	2nd	$G(b_0b_1)$	$G(b_2b_3)$	$G(b_4b_5)$	$G(b_6b_7)$	$G(B_2B_3)$	$G(B_2B_3)$	$G(B_4B_5)$	$G(B_6B_7)$

NOTE:

G: indicates that gray shade is applied to the pair
SS: Single Scan and DS: Double Scan

[1] $G(A_7A_6)$, $G(A_5A_4)$, ... can be easily obtained by swapping the contents of LCD_GRAY1 and LCD_GRAY2 registers

Advantage: Displayed byte is the same order as the stored byte
Disadvantage: Complex and slow data shifts and bit index calculations.

Shift Operations on the Displayed Data

To better understand how shift operations work in both modes, the following section goes through a Logical Shift Left 1 on a 32-bit word of frame buffer 1. Starting with a 32-bit word:

$D_7D_6D_5D_4D_3D_2D_1D_0C_7C_6C_5C_4C_3C_2C_1C_0B_7B_6B_5B_4B_3B_2B_1B_0A_7A_6A_5A_4A_3A_2A_1A_0$

Prior to a shift operation the data will be displayed on the LCD Panel as follows (assume a SS, Binary mode):

Format #0

$A_0A_1A_2A_3A_4A_5A_6A_7B_0B_1B_2B_3B_4B_5B_6B_7C_0C_1C_2C_3C_4C_5C_6C_7D_0D_1D_2D_3D_4D_5D_6D_7$.....

Format #1

$A_7A_6A_5A_4A_3A_2A_1A_0B_7B_6B_5B_4B_3B_2B_1B_0C_7C_6C_5C_4C_3C_2C_1C_0D_7D_6D_5D_4D_3D_2D_1D_0$.....

After a shift operation the shifted word will look like:

$D_6D_5D_4D_3D_2D_1D_0C_7C_6C_5C_4C_3C_2C_1C_0B_7B_6B_5B_4B_3B_2B_1B_0A_7A_6A_5A_4A_3A_2A_1A_00$

and the frame buffer in memory will look like:

Table 10-30.
Frame Buffer 1 After Shift Operation

BYTE ADDRESS	DATA
:	:
3	$D_6D_5D_4D_3D_2D_1D_0C_7$
2	$C_6C_5C_4C_3C_2C_1C_0B_7$
1	$B_6B_5B_4B_3B_2B_1B_0A_7$
0	$A_6A_5A_4A_3A_2A_1A_00$

and the LCD Panel will look like:

Format #0

$0A_0A_1A_2A_3A_4A_5A_6A_7B_0B_1B_2B_3B_4B_5B_6B_7C_0C_1C_2C_3C_4C_5C_6C_7D_0D_1D_2D_3D_4D_5D_6$.....

Format #1

$A_6A_5A_4A_3A_2A_1A_00B\ _6B_5B_4B_3B_2B_1B_0A_7C_6C_5C_4C_3C_2C_1C_0B_7D_6D_5D_4D_3D_2D_1D_0C_7$.....

For Format 0, the shifted 0 was displayed on the left and all pixels shifted one position to the right. In Format 1, the shifted 0 was displayed between the A_0 and B_6.

COLOR SUPPORT

Passive color panels require the same 790A control signals as monochrome panels (S, CP1, CP2). Each color pixel consists of 3 bits (Red, Green, Blue), so each byte will consist of two 2/3 color pixels (Table 10-30). The 790A can be programmed to interface to passive color panels by treating each color bit as a monochrome pixel (1 color pixel = 3 monochrome pixels). For example, a QVGA color panel consists of 320 × RGB × 240. By treating the panel as a 960 × 240 monochrome panel, the 790 LCD registers can be easily programmed to interface to the color panel (examples: Tables 10-52, 10-53).

Table 10-31.
Passive Color Frame Buffer (RGB/Pixel)

BYTE ADDRESS	COLOR DISPLAY DATA							
	D7	D6	D5	D4	D3	D2	D1	D0
:	:	:	:	:	:	:	:	:
3	G10	R10	B9	G9	R9	B8	G8	R8
2	B7	G7	R7	B6	G6	R6	B5	G5
1	R5	B4	G4	R4	B3	G3	R3	B2
0	G2	R2	B1	G1	R1	B0	G0	R0

BASIC TIMING

The LCD controller generates control signals (CP1,CP2,S,MCLK) for the LCD panel based on:

1. System Clock from Power Management Unit, LCD_in_CLK
2. Control Registers:
   - LCD_MODE
   - LCD_BC
   - LCD_CP1W
   - LCD_DUTY
   - LCD_CLKDIV
   - LCD_MCLKW
3. External Memory configuration and Speed

The designer can program the above registers according to the LCD panel specifications. The following section will detail the relationship between the control registers, CP1, CP2, S, and MCLK output signals.

NOTE: A separate Application Note will detail how to best utilize the 790A Unified Memory Architecture, frame buffer data flow and reduce end of frame delay.

Timing Equations

Figure 10-13 shows a general LCD Controller timing diagram. The following table describes the parameters used in the timing equations and diagrams:

Table 10-32.
Parameter Description

PARAMETER	DESCRIPTION
DUTY[1]	Number of CP1 Pulses per Frame (LCD_DUTY)
BC[1]	Number of Memory Bytes in a Horizontal Line (LCD_BC)
CP1W[1]	Line Pulse High Width (LCD_CP1W)
CLKDIV[1]	Clock Frequency Divider (LCD_CLKDIV)
$t_{LCD\_in\_CLK}$	LCD Input Clock from Power Management Unit
t_{lcdclk}	LCD Reference Clock (Output of Clock Divider)
t_S	Frame Pulse Period
t_{CP1}	Line Pulse Period
t_{CP2}	Shift Clock Period
t_{PXFR}	Pixels Transfer Time per Line
t_{CP1W}	Line Pulse High Width Time
t_{12}	Current Frame CP1 ↓ to Current Frame CP2 ↑
t_{12F}	Current Frame CP1 ↓ to next frame CP2 ↑
t_{21}	CP2 ↓ to CP1 ↑
t_{DS}	Data Setup time
t_{DH}	Data Hold time
t_{SS}	S signal Setup time
t_{SH}	S signal Hold time
t_{1M}	CP1 ↓ to MCLK Inverting

The following equations[1] and parameters describe the relationship between LCD input Clock, S, CP1, CP2, and MCLK.

$t_{lcdclk} = CLKDIV \times t_{LCD\_in\_CLK}$ ($t_{LCD\_in\_CLK}$ = XCLK period)

$t_S = t_{CP1} \times DUTY$

$t_{CP1} = t_{PXFR} + t_{CP1W} + t_{12}$

NOTE :

1. Decimal "equivalent" values, as defined in the corresponding parameters tables, must be used in timing equations. As an example, if LCD_CLKDIV is programmed to 0x00, the decimal "equivalent" is 2 (from Table 10-26).

t_{PXFR}, t_{CP1W}, and t_{CP2} vary from one display mode to another as follows:

Table 10-33.
t_{PXFR}, t_{CPIW} and t_{CP2} Parameters

DISPLAY MODE	t_{PXFR}	t_{CP1W}	t_{CP2}
1a (4-bit)	$2 \times BC \times t_{CP2}$	$(CP1W+1/2) \times t_{CP2}$	$2 \times t_{lcdclk}$
1b (8-bit)	$BC \times t_{CP2}$	$(CP1W+1/2) \times t_{CP2}$	$4 \times t_{lcdclk}$
2	$BC \times t_{CP2}$	$(CP1W+1/2) \times t_{CP2}$	$4 \times t_{lcdclk}$
3a (4-bit)	$2 \times BC \times t_{CP2}$	$(CP1W-1/2) \times t_{CP2}$	$4 \times t_{lcdclk}$
3b (8-bit)	$BC \times t_{CP2}$	$(CP1W) \times t_{CP2}$	$8 \times t_{lcdclk}$
4	$BC \times t_{CP2}$	$(CP1W+1/2) \times t_{CP2}$	$8 \times t_{lcdclk}$
5	$2 \times BC \times t_{CP2}$	$(CP1W-1/2) \times t_{CP2}$	$4 \times t_{lcdclk}$
6	$BC \times t_{CP2}$	$(CP1W+1/2) \times t_{CP2}$	$8 \times t_{lcdclk}$

Other LCD timing parameters are shown in Table 10-34.

Table 10-34.
Other LCD Timing Parameters

VARIABLE	VALUE
t_{DS}	$1/2 \times t_{CP2}$
t_{DH}	$1/2 \times t_{CP2}$
t_{12}	$1/2 \times t_{CP2}$
t_{12F}	Variable[2]
t_{21}	$1/2 \times t_{CP2}$
t_{SH}	$3.5 \times t_{CP2}$
t_{SS}	$t_{PXFR} + t_{CP1W} - t_{SH}$
t_{1M}[3]	0

NOTES:

1. Actual timing may vary from those calculated depending on current instruction executed, memory speed, DRAM Refresh Rate, . . . etc.
2. Since this delay happens once a frame, its effect on the frame rate is small, t_{12} will be used in the timing equations.
3. MCLK clock changes on a falling edge of CP1 clock as programmed by LCD_MCLKW register.

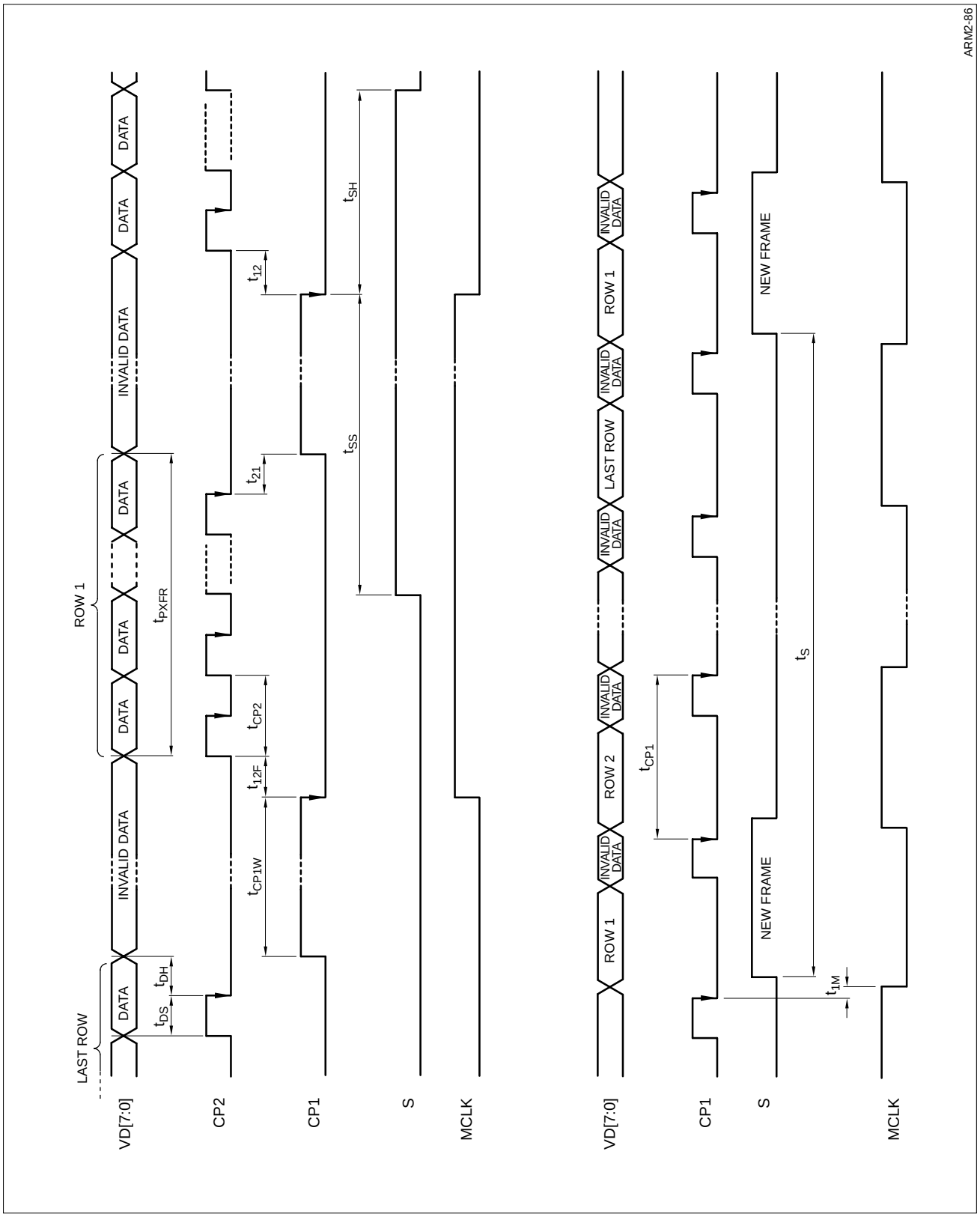


Figure 10-13. MCLK and CP1 Relationship

FRAME RATE CALCULATIONS

Using the eight modes of operations listed in Table 10-14, the LCD controller parameters, and the timing equations listed in the previous section, a maximum frame rate can be calculated for the 790A (25 MHz and 16.7 MHz) driving a sample of panel resolutions as shown in Table 10-35.

Table 10-35.
790A Max. Frame Rates for VGA/HVGA/QVGA LCD Panels

MODE	FEATURES[1]	MAXIMUM FRAME RATE (HZ) (XCLK = 25 MHz @ 5 V/16.7 MHz @ 3.3 V]		
		VGA (640 × 480)	HVGA (480 × 320)	QVGA (320 × 240)
1a	SS, B, 4-bit, DIV	Note 2	156/104	307/205
1b	SS, B, 8-bit, DIV	Note 2	155/104	Note 3
2	SS, G, 4-bit, DIV	Note 2	79*/53*	157/105*
3a	SS, B, 4-bit, OR	Note 2	78/52*	151/101
3b	SS, B, 8-bit, OR	Note 2	78/52*	Note 3
4	SS, G, 4-bit, OR	Note 2	40*/27*	78*/52*
5	DS, B, 4-bit	78/52*	Note 4	Note 4
6	DS, G, 4-bit	40*/27*	Note 4	Note 4

NOTES:

1. SS: Single Scan DS: Dual Scan
   B: Binary G: Gray
   OR: OR function DIV: Panel Division
   4-bit: 4 bits transferred per CP2
   8-bit: 8 bits transferred per CP2
   LCD_DUTY = The number of rows in Single Scan mode
   LCD_DUTY = One Half the number of rows in Dual Scan mode
   The minimum legal values for LCD_CP1W & LCD_CLKDIV are used
2. VGA panels are typically Dual Scan
3. QVGA panels typically use 4-bits per CP2
4. HVGA and lower resolution panels are typically Single Scan
5. *Depending on the LCD Panel, the frame rate can be out of specification and/or cause flickering.

PROGRAMMING EXAMPLES

This section will show examples on how to program the LCD controller to operate in each of the 8 modes of operation. In each mode, the LCD panel specifications are stated, and the LCD controller registers' values are calculated. A sample of the calculations will be shown for the first mode of operation. The rest of the modes will only show a table summarizing the LCD controller registers' values but will follow the same calculations as the first mode.

Mode 1a:

Table 10-36.
LCD Specifications (e.g. Sharp's LM32P07)

PARAMETER	RANGE
Resolution	320 (H) × 240 (V)
Duty Cycle	1/240
Frame Rate	12.5 ms - 14.3 ms
Transfer Clock Period	≥ 300 ns
Data Bits	4
Scan	Single
XCLK	24 MHz

LCD_MODE

From Table 10-14, LCD_MODE is set to 0x83.

LCD_BC

LCD_BC is calculated from the horizontal resolution of the LCD panel. Since this is a binary mode, LCD_BC = (320 pixels/line) × (1 bit/pixel) × (1 Byte)/(8 bits) = 40 bytes/line. Referring to Table 10-15, this translates to LCD_BC set to 0x27.

LCD_DUTY

LCD_DUTY is calculated from the duty cycle of the LCD panel. The duty cycle is specified as 1/240. Using Table 10-18, LCD_DUTY[9:0] is set to 0b0011101111.

LCD_CLK

LCD_CLK is calculated from the transfer clock period specified by the LCD panel as follows:

1. The transfer clock period, t_{CP2}, is specified in Table 10-36 as a minimum of 300 ns.
2 From Table 10-32a, $t_{CP2} = 2 \times t_{lcdclk} = 2 \times CLKDIV \times t_{LCD\_in\_CLK}$
3. Since $t_{CP2} \geq 300$ ns, $2 \times CLKDIV \times 41.67$ ns ≥ 300 ns
4. Solving for CLKDIV, CLKDIV ≥ 3.6. Referring to Table 10-26, the closest decimal value would be 4 which would translate to LCD_CLK set to 0x02.
5. Calculating $t_{CP2} = 2 \times 4 \times 41.67$ ns = 333.36 ns

LCD_CP1W

LCD_CP1W (a.k.a CP1W) is calculated from the equations in the 'Basic Timing' section and Table 10-33. For mode 1a:

$t_{CP1} = t_S$ / DUTY ------- (1)

$t_{CP1} = t_{PXFR} + t_{CP1W} + t_{12} = 2 \times BC \times t_{CP2} + (CP1W+1/2) \times t_{CP2} + 1/2 \times t_{CP2}$ -----(2)

All of the variables have been given or calculated with the exception of CP1W. Combining equations (1) and (2) and selecting a frame rate, t_S, of 13 ms as given in Table 10-36:

$13000000/240 = 2 \times 40 \times 333.36 + (CP1W+1/2) \times 333.36 + 333.36/2$

$CP1W \approx 82 (\geq 2)$

Referring to Table 10-17, this would translate to LCD_CP1W set to 0x29.

Using the new CP1W value and equations (1) and (2) to calculate the frame rate will produce a rate of 13.04 ms which is within the range of the LCD specifications in Table 10-36.

The previous calculations were based on minimizing t_{CP2} which resulted in a large CP1 high pulse (t_{CP1W}). Another approach would be to minimize CP1 high pulse which would result in large t_{CP2} and a larger clock divider (LCD_CLKDIV)

Mode 1a Results:

Table 10-37.
LCD Registers' Value in Mode 1a Example

REGISTER NAME	REGISTER VALUE	COMMENTS
LCD_MODE[7:0]	0x83	Binary, 4-bit, Division, Single
LCD_BC[7:0]	0x27	Byte Count = 40
LCD_CP1W[7:0]	0x29	CP1 Pulse Width = 82
LCD_DUTY[7:0]	0xEF	Duty Cycle = 1/240
LCD_DUTY[9:8]	0b00	
LCD_SADR1[7:0]	See Note 9.	
LCD_SADR1[15:8]		
LCD_SADR1[23:16]		
LCD_SADR1[31:24]		
LCD_SADR2[7:0]	See Note 10.	
LCD_SADR2[15:8]		
LCD_SADR2[23:16]		
LCD_SADR2[31:24]		
LCD_VLC1[7:0]	See Note 4.	
LCD_VLC1[9:8]		
LCD_VDLT[7:0]	See Note 8.	
LCD_GRAY1[7:0]	X	Don’t Care
LCD_GRAY2[7:0]	X	Don’t Care
LCD_CLKDIV[7:0]	0x02	Clock Divider = 4
LCD_MCLKW[7:0]	See Note 5.	
LCD_MCLKW[9:8]		
FRAME RATE	13.04 ms	

NOTE: Programming LCD_CP1W to 0x02 and LCD_CKKDIV to 0x06 will produce a frame rate period of 13.6 ns.

Mode 1b:

Table 10-38.
LCD Specifications (e.g. Epson's EG9013F-NZ-1)

PARAMETER	RANGE
Resolution	640 (H) × 480 (V)
Duty Cycle	1/480
Frame Rate	12 ms - 15 ms
Transfer Clock Period	≥ 160 ns
Data Bits	8
Scan	Single
XCLK	24 MHz

Mode 1b Results:

Table 10-39.
LCD Registers' Value in Mode 1b Example

REGISTER NAME	REGISTER VALUE	COMMENTS
LCD_MODE[7:0]	0x87	Binary, 8-bit, Division, Single
LCD_BC[7:0]	0x4F	Byte Count = 80
LCD_CP1W[7:0]	0x01	CP1 Pulse Width = 2
LCD_DUTY[7:0]	0xDF	Duty Cycle = 1/480
LCD_DUTY[9:8]	0b01	
LCD_SADR1[7:0]	See Note 9.	
LCD_SADR1[15:8]		
LCD_SADR1[23:16]		
LCD_SADR1[31:24]		
LCD_SADR2[7:0]	See Note 10.	
LCD_SADR2[15:8]		
LCD_SADR2[23:16]		
LCD_SADR2[31:24]		
LCD_VLC1[7:0]	See Note 4.	
LCD_VLC1[9:8]		
LCD_VDLT[7:0]	See Note 8.	
LCD_GRAY1[7:0]	X	Don't Care
LCD_GRAY2[7:0]	X	Don't Care
LCD_CLKDIV[7:0]	0x00	Clock Divider = 2
LCD_MCLKW[7:0]	See Note 5.	
LCD_MCLKW[9:8]		
FRAME RATE	13.28 ms	

Mode 2:

Table 10-40.
LCD Specifications (e.g. Sharp's LM32P07)

PARAMETER	RANGE
Resolution	320 (H) × 240 (V)
Duty Cycle	1/240
Frame Rate	12.5 ms - 14.3 ms
Transfer Clock Period	≥ 300 ns
Data Bits	4
Scan	Single
XCLK	24 MHz

Mode 2 Results:

Table 10-41.
LCD Registers' Value in Mode 2 Example

REGISTER NAME	REGISTER VALUE	COMMENTS
LCD_MODE[7:0]	0x8B	Gray, 4-bit, Division, Single
LCD_BC[7:0]	0x4F	Byte Count = 80
LCD_CP1W[7:0]	0x01	CP1 Pulse Width = 2
LCD_DUTY[7:0]	0xEF	Duty Cycle = 1/240
LCD_DUTY[9:8]	0b00	
LCD_SADR1[7:0]	See Note 9.	
LCD_SADR1[15:8]		
LCD_SADR1[23:16]		
LCD_SADR1[31:24]		
LCD_SADR2[7:0]	See Note 10.	
LCD_SADR2[15:8]		
LCD_SADR2[23:16]		
LCD_SADR2[31:24]		
LCD_VLC1[7:0]	See Note 4.	
LCD_VLC1[9:8]		
LCD_VDLT[7:0]	See Note 8.	
LCD_GRAY1[7:0]	GRAY1	
LCD_GRAY2[7:0]	GRAY2	
LCD_CLKDIV[7:0]	0x00	Clock Divider = 2
LCD_MCLKW[7:0]	See Note 5.	
LCD_MCLKW[9:8]		
FRAME RATE	6.64 ms	To eliminate flickering, the LM32P07 had to be refreshed at higher rates that its spec.

Mode 3a:

Table 10-42.
LCD Specifications (e.g. Sharp's LM32P07)

PARAMETER	RANGE
Resolution	320 (H) × 240 (V)
Duty Cycle	1/240
Frame Rate	12.5 ms - 14.3 ms
Transfer Clock Period	≥ 300 ns
Data Bits	4
Scan	Single
XCLK	24 MHz

Mode 3a Results:

Table 10-43.
LCD Registers' Value in Mode 2 Example

REGISTER NAME	REGISTER VALUE	COMMENTS
LCD_MODE[7:0]	0x93	Binary, 4-bit, OR, Single
LCD_BC[7:0]	0x27	Byte Count = 40
LCD_CP1W[7:0]	0x29	CP1 Pulse Width = 82
LCD_DUTY[7:0]	0xEF	Duty Cycle = 1/240
LCD_DUTY[9:8]	0b00	
LCD_SADR1[7:0]	See Note 9.	
LCD_SADR1[15:8]		
LCD_SADR1[23:16]		
LCD_SADR1[31:24]		
LCD_SADR2[7:0]	See Note 10.	
LCD_SADR2[15:8]		
LCD_SADR2[23:16]		
LCD_SADR2[31:24]		
LCD_VLC1[7:0]	X	
LCD_VLC1[9:8]		
LCD_VDLT[7:0]	See Note 8.	
LCD_GRAY1[7:0]	X	
LCD_GRAY2[7:0]	X	
LCD_CLKDIV[7:0]	0x00	Clock Divider = 2
LCD_MCLKW[7:0]	See Note 5.	
LCD_MCLKW[9:8]		
FRAME RATE	12.96 ms	

NOTE: Programming LCD_CP1W to 0x03 and LCD_CLKDIV to 0x02 will produce a frame rate period of 13.76 ns.

Mode 3b:

Table 10-44.
LCD Specifications

PARAMETER	RANGE
Resolution	480 (H) × 320 (V)
Duty Cycle	1/320
Frame Rate	8 ms - 16.9 ms
Transfer Clock Period	≥ 160 ns
Data Bits	8
Scan	Single
XCLK	24 MHz

Mode 3b Results:

Table 10-45.
LCD Registers' Value in Mode 3b Example

REGISTER NAME	REGISTER VALUE	COMMENTS
LCD_MODE[7:0]	0x97	Binary, 8-bit, OR, Single
LCD_BC[7:0]	0x3B	Byte Count = 60
LCD_CP1W[7:0]	0x03	CP1 Pulse Width = 6
LCD_DUTY[7:0]	0x3F	Duty Cycle = 1/320
LCD_DUTY[9:8]	0b01	
LCD_SADR1[7:0]	See Note 9.	
LCD_SADR1[15:8]		
LCD_SADR1[23:16]		
LCD_SADR1[31:24]		
LCD_SADR2[7:0]	See Note 10.	
LCD_SADR2[15:8]		
LCD_SADR2[23:16]		
LCD_SADR2[31:24]		
LCD_VLC1[7:0]	X	
LCD_VLC1[9:8]		
LCD_VDLT[7:0]	See Note 8.	
LCD_GRAY1[7:0]	X	
LCD_GRAY2[7:0]	X	
LCD_CLKDIV[7:0]	0x00	Clock Divider = 2
LCD_MCLKW[7:0]	See Note 5.	
LCD_MCLKW[9:8]		
FRAME RATE	14.19 ms	

Mode 4:

Table 10-46.
LCD Specifications (e.g. Sharp's LM32P07)

PARAMETER	RANGE
Resolution	320 (H) × 240 (V)
Duty Cycle	1/240
Frame Rate	12.5 ms - 14.3 ms
Transfer Clock Period	≥ 300 ns
Data Bits	4
Scan	Single
XCLK	24 MHz

Mode 4 Results:

Table 10-47.
LCD Registers' Value in Mode 4 Example

REGISTER NAME	REGISTER VALUE	COMMENTS
LCD_MODE[7:0]	0x9B	Gray, 4-bit, OR, Single
LCD_BC[7:0]	0x4F	Byte Count = 80
LCD_CP1W[7:0]	0x01	CP1 Pulse Width = 2
LCD_DUTY[7:0]	0xEF	Duty Cycle = 1/240
LCD_DUTY[9:8]	0b00	
LCD_SADR1[7:0]	See Note 9.	
LCD_SADR1[15:8]		
LCD_SADR1[23:16]		
LCD_SADR1[31:24]		
LCD_SADR2[7:0]	See Note 10.	
LCD_SADR2[15:8]		
LCD_SADR2[23:16]		
LCD_SADR2[31:24]		
LCD_VLC1[7:0]	X	
LCD_VLC1[9:8]		
LCD_VDLT[7:0]	See Note 8.	
LCD_GRAY1[7:0]	GRAY1	
LCD_GRAY2[7:0]	GRAY2	
LCD_CLKDIV[7:0]	0x00	Clock Divider = 2
LCD_MCLKW[7:0]	See Note 5.	
LCD_MCLKW[9:8]		
FRAME RATE	13.28 ms	*LM32P07 will exhibit flickering. * This is the max. frame rate supported by 790A running at 24 MHz.

Mode 5:

Table 10-48.
LCD Specifications (e.g. Sharp's LM64K101)

PARAMETER	RANGE
Resolution	640 (H) × 480 (V)
Duty Cycle	1/240
Frame Rate	8 ms - 16.9 ms
Transfer Clock Period	≥ 230 ns
Data Bits	8
Scan	Dual
XCLK	24 MHz

Mode 5 Results:

Table 10-49.
LCD Registers' Value in Mode 5 Example

REGISTER NAME	REGISTER VALUE	COMMENTS
LCD_MODE[7:0]	0xA3	Binary, Dual
LCD_BC[7:0]	0x4F	Byte Count = 80
LCD_CP1W[7:0]	0x03	CP1 Pulse Width = 6
LCD_DUTY[7:0]*	0xF0	Duty Cycle = 1/240 Requires 241 CP1 Pulses
LCD_DUTY[9:8]	0b00	
LCD_SADR1[7:0]	See Note 9.	
LCD_SADR1[15:8]		
LCD_SADR1[23:16]		
LCD_SADR1[31:24]		
LCD_SADR2[7:0]	See Note 10.	
LCD_SADR2[15:8]		
LCD_SADR2[23:16]		
LCD_SADR2[31:24]		
LCD_VLC1[7:0]	X	
LCD_VLC1[9:8]		
LCD_VDLT[7:0]	See Note 8.	
LCD_GRAY1[7:0]	X	
LCD_GRAY2[7:0]	X	
LCD_CLKDIV[7:0]	0x00	Clock Divider = 2
LCD_MCLKW[7:0]	See Note 5.	
LCD_MCLKW[9:8]		
FRAME RATE	13.34 ms	

NOTE: *LM64K101 required 241 CP1 pulses to operate properly.

Mode 6:

Table 10-50.
LCD Specifications

PARAMETER	RANGE
Resolution	480 (H) × 320 (V)
Duty Cycle	1/162
Frame Rate	8 ms - 16.9 ms
Transfer Clock Period	≥ 160 ns
Data Bits	8
Scan	Dual
XCLK	24 MHz

Mode 6 Results:

Table 10-51.
LCD Registers' Value in Mode 6 Example

REGISTER NAME	REGISTER VALUE	COMMENTS
LCD_MODE[7:0]	0xAB	Gray, Dual
LCD_BC[7:0]	0x77	Byte Count = 120
LCD_CP1W[7:0]	0x04	CP1 Pulse Width = 8
LCD_DUTY[7:0]	0xA1	Duty Cycle = 1/162
LCD_DUTY[9:8]	0b00	
LCD_SADR1[7:0]	See Note 9.	
LCD_SADR1[15:8]		
LCD_SADR1[23:16]		
LCD_SADR1[31:24]		
LCD_SADR2[7:0]	See Note 10.	
LCD_SADR2[15:8]		
LCD_SADR2[23:16]		
LCD_SADR2[31:24]		
LCD_VLC1[7:0]	X	
LCD_VLC1[9:8]		
LCD_VDLT[7:0]	See Note 8.	
LCD_GRAY1[7:0]	GRAY1	
LCD_GRAY2[7:0]	GRAY2	
LCD_CLKDIV[7:0]	0x00	Clock Divider = 2
LCD_MCLKW[7:0]	See Note 5.	
LCD_MCLKW[9:8]		
FRAME RATE	13.93 ms	

PASSIVE COLOR PANEL EXAMPLE

Table 10-52.
LCD Specification

PARAMETER	RANGE
Resolution	320 (H) × RGB × 240 (V)
Frame Rate	8 ms – 11 ms
Duty Cycle	1/240
Transfer Clock	≥ 80 ns
Data Bits	8
Scan	Single
XCLK	24 MHz

Table 10-53.
LCD Registers' Value

REGISTER NAME	REGISTER VALUE	COMMENTS
LCD_MODE[7:0]	0x87	Binary, 8-bit, Division, Single
LCD_BC[7:0]	0x77	Byte Count = 120 (320 × 3/8)
LCD_CP1W[7:0]	0x01	CP1 Pulse Width = 2
LCD_DUTY[7:0]	0xEF	Duty Cycle = 1/240
LCD_DUTY[9:8]	0b00	
LCD_SADR1[7:0]	See Note 9.	
LCD_SADR1[15:8]		
LCD_SADR1[23:16]		
LCD_SADR1[31:24]		
LCD_SADR2[7:0]	See Note 10.	
LCD_SADR2[15:8]		
LCD_SADR2[23:16]		
LCD_SADR2[31:24]		
LCD_VLC1[7:0]	X	
LCD_VLC1[9:8]		
LCD_VDLT[7:0]	See Note 8.	
LCD_GRAY1[7:0]	X	
LCD_GRAY2[7:0]	X	
LCD_CLKDIV[7:0]	0x00	Clock Divider = 2
LCD_MCLKW[7:0]	See Note 5.	
LCD_MCLKW[9:8]		
FRAME RATE	9.84 ms	

NOTES:

1. Calculations will be based on the equations and tables in 'Basic Timing' section of this chapter. The system clock, XCLK, is 24 MHz (41.67 ns). This means that $t_{LCD\_in\_CLK}$ = 41.67 ns when the LCD controller clock is enabled via the CPMU. A frame rate value will be selected from the LCD specifications.
2. LCD_BC (a.k.a. BC) is a function of either binary mode or gray mode. Transfer size, OR, and dual/single scan have no affect on BC.
3. LCD_CP1W (a.k.a CP1W) must always be ≥ 2. If not, calculations must be redone with a different set of parameters (e.g. choose a slower frame rate or a faster CP2 rate).
4. LCD_VLC1 (a.k.a VLC1) is only used in single scan, division (OR = 0) modes (Modes 1a, 1b, 2). To get a single screen, make LCD_VLC1 ≥ LCD_DUTY. For a divided screen, make LCD_VLC1 ≥ LCD_DUTY. For other modes, it is a don't care.
5. LCD panel is assumed to have an internal MCLK generator, so LCD_MCLK will be a Don't Care in the examples.
6. LCD_GRAY1 and LCD_GRAY2 are only required in gray mode operations (Modes 2, 4, 6). The value of each register will depend on the LCD panel characteristics. The tables for the above modes will reflect GRAY1 and GRAY2 for the two gray shades.
7. In all modes, LCDC bit will be '1', REV will be '0', and DISP will be '1' and Don't Cares in OR and XSIZE will be converted to '0's in LCD_MODE.
8. LCD_VDLT is set to '0x00' in all examples (no virtual screens).
9. LCD_SADR1 will hold the starting address of the frame buffer for screen #1.
10. LCD_SADR2 will hold the starting address of the frame buffer for screen #2.
11. The frame buffer should always reside in external memory. The frame buffer segment should have at least a system read privilege and should not be cacheable. The frame buffer should be large enough to hold the image and the extra lines in the case where the duty cycle is larger than the number of vertical lines.
12. The decimal values and their binary/hex equivalent are not a one to one map.
13. X = Don't Care

Chapter 11

COUNTERS/TIMERS

INTRODUCTION

The 790A provides three independent 16-bit Counter/Timer channels. Each channel (82C54 compatible) can operate in one of six modes and work with either Binary or BCD Counting.

FEATURES

- Three Independent 8-Bit/16-Bit Programmable Counter/Timer
- Six Modes of Operation:
  - Mode 0: Interrupt on Terminal Count
  - Mode 1: Hardware Retriggerable One-Shot
  - Mode 2: Rate Generator
  - Mode 3: Square Wave Generator
  - Mode 4: Software Triggered Strobe
  - Mode 5: Hardware Triggered Strobe
- Binary or BCD Counting
- ARM7DI has access to the following:
  - Current Counter Value
  - Control Word
  - Current State of OUT signals
- Selectable Input Clock
  - System Clock
  - External Clock
- Power Management
  - Scale Down System Clock
  - Halt Input Clock

BLOCK DIAGRAM

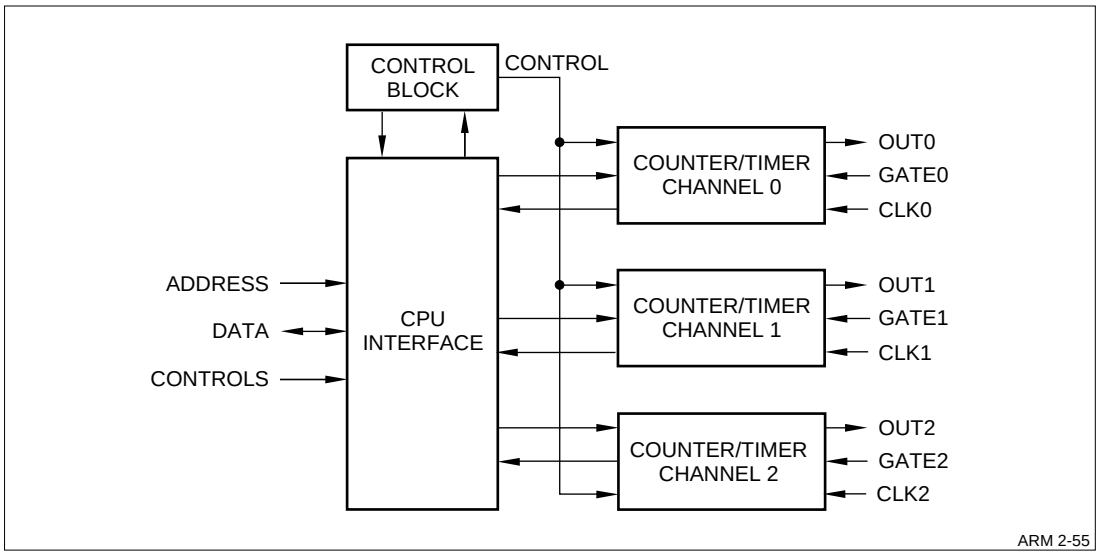


Figure 11-1. Three Channel Counter/Timer Block Diagram

INTERNAL DIAGRAM

The following diagram shows the internals of one of the three Counter/Timers. The Control Word Register is common to all three Counter/Timers.

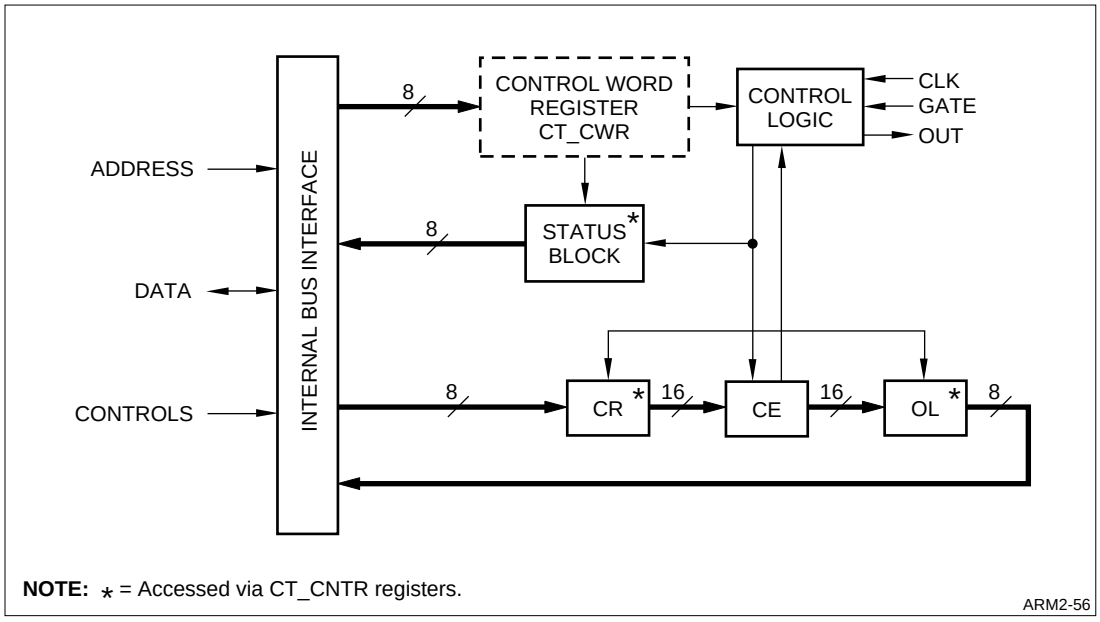


Figure 11-2. Internal Diagram

EXTERNAL INTERFACE

Table 11-1.
Counter/Timer External Interface

NAME	TYPE	DESCRIPTION
CTOUTi	Output	Outputs of Counter/Timers (i = 0, 1, 2)
CTGATEi	Input	Gates of Counter/Timers (I = 0, 1, 2)
CTCLK	Input	Counter/Timers External Clock. One external clock for all three counters. This clock is multiplexed with an internally generated clock based on the system clock XCLK

REGISTER MAP

The base address for the Counter/Timers is 0xFFFF1800. The offset, initial value, access and number of bits for each register are as follows:

Table 11-2.
Counter/Timer Registers

COUNTER/TIMER REGISTER	ADDRESS OFFSET[7:0]	RESET VALUE	ACCESS	NUMBER OF BITS
CT_CNTR0	0x00	undefined	R/W	8
CT_CNTR1	0x04	undefined	R/W	8
CT_CNTR2	0x08	undefined	R/W	8
CT_CWR	0x0C	undefined	W	8

GENERAL OPERATION

The three counters/timer channels (a.k.a counters) are identical but operate independently. A block diagram for one of the counters is shown above.

The three counters share one Control Word register (CT_CWR), shown in dashed lines in the internal diagram. The control word register selects/defines the mode of operation for each counter, and allows access to the counter value, and counter status.

The status logic is used to latch the contents of the control word register, counter and the output CTOUT.

The actual counter is labeled CE (Counting Element). It is a 16-bit synchronous down counter. The CE can't be accessed directly. An Output Latch (OL) and a Counter Register (CR) are used to read and write CE respectively.

OL consists of two 8-bit latches: OLM and OLL. OLM is the most significant byte and OLL is the least significant byte. These latches normally follow CE, until a CounterLatch or a ReadBack command is received. These commands latch the value of the CE into OL. OL is read one byte at a time via CT_CNTRi.

CR consists of two 8-bit latches: CRM and CRL. CRM is the most significant byte and CRL is the least significant byte. When a new count is written into the counter (via CT_CNTRi), the count is stored in CR and then transferred into CE. CR is written one byte at a time.

REGISTER DESCRIPTION

Count Registers (CT_CNTR0, CT_CNTR1, CT_CNTR2)

7	6	5	4	3	2	1	0
Counter Value							

The count registers are used in conjunction with the control word register to indirectly read and write the three counting elements (CEs) and read the counter(s) status. A read access will read the contents of the Output Latch (OL) or latched status and a write access will write the Counter Register (CR).

Control Word Register (CT_CWR)

7	6	5	4	3	2	1	0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

The control word register is a write only register. The contents of the register are available through CommandLatch or ReadBack commands.

Table 11-5.
Control Word Fields

BIT POSITION	NAME	DESCRIPTION
0	BCD	Binary Coded Decimal: 0 ---> Binary Counter, 16-bit 1 ---> BCD Counter, 4 Decades If a counter is programmed to operate using one byte only, the other byte is forced to zero. If a counter is set into BCD mode, it must be programmed with BCD values only (0-9), otherwise results will be unpredictable.
3:1	M2:M0	Counter Mode: 000 ---> Mode 0: Interrupt On Terminal Count 001 ---> Mode 1: Hardware Retriggerable One-Shot 010 ---> Mode 2: Rate Generator 011 ---> Mode 3: Square Wave Generator 100 ---> Mode 4: Software Triggered Strobe 101 ---> Mode 5: Hardware Triggered Strobe
5:4	RW1:RW0	Read/Write 00 ---> CounterLatch Command 01 ---> Read/Write Least Significant Byte (LSB) only 10 ---> Read/Write Most Significant Byte (MSB) only 11 ---> Read/Write LSB followed by MSB (accessing the two bytes need not be back to back. Read, Write, or Programming operations to other counters may be inserted between them)
7:6	SC1:SC0	Select Counter 00 ---> Select Counter 0 01 ---> Select Counter 1 10 ---> Select Counter 2 11 ---> ReadBack Command

WRITE OPERATIONS

Programming the counters requires following two constraints:

1. For each counter, the control word register (CT_CWR) must be written before the initial count is written. The control word register contains the address of the counter to which it applies.
2. The count written to the counter should follow the format defined in the control word register (RW1 and RW0 fields).

An initial value can be written to a counter any time and will not change the counter's mode of operation but counting may be affected by the new count value as described later.

READ OPERATIONS

There are three possible methods for reading the counter values:

1. A simple read operation
2. The CounterLatch Command
3. The ReadBack Command

Simple Read Operation

This is a simple read of the counter (CT_CNTR0, CT_CNTR1, or CT_CNTR2). In order to get a valid result, the clock input to the counter should be disabled by either the GATE input or stopping the clock. This will guarantee that the counter is not changing while a read operation is performed.

Counter Latch Command

7	6	5	4	3	2	1	0
SC1	SC0	0	0	0	0	0	0

Control Word Countents for Counter Latch Command

A CounterLatch command will latch the state of the counter selected. This command uses the control word register (CT_CWR) and is selected when RW1 and RW0 bits are both zero. SC1 and SC0 are used to select the counter to latch. The effect of the command is that the counter value at the time of the command, is latched into Output Latch (OL) and preserved until the latched value is read via its CT_CNTR or the counter is reprogrammed.

If a subsequent CounterLatch command is issued to the same counter before the result of the first command has been read, it will have no effect on the original latched value. Once the counter is read according to the programmed format, the OL will resume tracking the Counting Element (CE) and another CounterLatch command can take place for the counter. Multiple CounterLatch commands can be issued independently to multiple counters to latch their count values. CounterLatch commands do not affect the counter's programmed mode.

Reads and writes of the same counter may be interwoven. For example, if the counter is programmed for two-byte counts, the following sequence is valid:

1. Read least significant byte
2. Write new least significant byte
3. Read most significant byte
4. Write new most significant byte

ReadBack Command

7	6	5	4	3	2	1	0
1	1	$\overline{\text{COUNT}}$	$\overline{\text{STATUS}}$	CNT2	CNT1	CNT0	0

Control Word Register Contents for ReadBack Command

A ReadBack command will either latch the state or status of the counter(s) selected. This command uses the control word register (CT_CWR) and is selected when SC1 and SC0 bits are both one. For the selected counter(s) the latched value is preserved until it is read back via its CT_CNTRi or the counter is reprogrammed. Each counter operates independently.

Table 11-8.
ReadBack Command Fields

BIT POSITION	NAME	DESCRIPTION
0	unused	Should be set to 0
1	CNT0*	Select Counter 0 0 ---> Counter 0 is not selected 1 ---> Counter 0 is selected
2	CNT1*	Select Counter 1 0 ---> Counter 1 is not selected 1 ---> Counter 1 is selected
3	CNT2*	Select Counter 2 0 ---> Counter 2 is not selected 1 ---> Counter 2 is selected
4	$\overline{\text{STATUS}}$	Status Latch 0 ---> Status of selected counter(s) are latched 1 ---> Status of selected counter(s) are not latched
5	$\overline{\text{COUNT}}$	Count Latch 0 ---> Count value of selected counter(s) are latched 1 ---> Count value of selected counter(s) are not latched

NOTE: More than one counter can be selected

As mentioned before, when $\overline{\text{STATUS}}$ is 0, the status of the selected counter(s) is latched at the time the command was issued. The format of the status byte for the selected counter(s) is as follows:

7	6	5	4	3	2	1	0
OUT	NULL COUNT	RW1	RW0	M2	M1	M0	BCD

Table 11-10.
Status Byte Fields

BIT POSITION	NAME	DESCRIPTION
0	BCD	Reflects the status of BCD bit as last programmed via the control word register (CT_CWR).
1	M0	Reflects the status of M0 bit as last programmed via the control word register (CT_CWR).
2	M1	Reflects the status of M1 bit as last programmed via the control word register (CT_CWR).
3	M2	Reflects the status of M2 bit as last programmed via the control word register (CT_CWR).
4	RW0	Reflects the status of RW0 bit as last programmed via the control word register (CT_CWR).
5	RW1	Reflects the status of RW1 bit as last programmed via the control word register (CT_CWR).
6	NULL COUNT	Reflects whether the last count value written to the counter via the Counter Register (CR) has been transferred to the Counting Element (CE) 0 ---> CR transferred to CE 1 ---> CR not transferred to CE Writing to the control word register (CT_CWR) or CR will cause NULL COUNT to be a one for the selected counter(s)
7	OUT	Reflects the status of the pin CTOUT

Multiple counters can have their status and/or count value latched with one ReadBack command. As is the case in the CounterLatch command, a second ReadBack command issued before the first one has been read or the counter is reprogrammed will have no effect. The value read is the result of the first ReadBack command. This is valid on a counter by counter basis. For example:

1. If a ReadBack command is issued on counters 1 and 2 to latch their count values
2. Then Counter 1 is read ---> counter 1 is unlatched
3. Then if a second ReadBack command is issued on counters 1 and 2, only counter 1 will be latched. Counter 2 latched value is unchanged.

If both the count value and status of a counter are latched, the first read operation of that counter (via CT_CNTRi) will return the latched status. The next one or two reads (depending on whether the counter is programmed for one or two byte-counts) return the latched count value.

COUNTER/TIMER CLOCKS

Referring to the internal block diagram, the clock driving the counter(s) (CLK) can either be an internally generated clock based on XCLK or an externally generated clock based on CTCLK. PCSR register in the Clock and Power Management Unit (CPMU) selects between the two clocks. Scaling down XCLK is controlled by CT0CCR, CT1CCR, and CT2CCR registers in the CPMU.

COUNTER/TIMER GATE

Referring to the internal block diagram, each Counter/Timer GATE input can be programmed to come from four different sources:

1. External Gate pin CTGATEi (x=0,1,2)
2. PWM_x Output (x=0,1,2)
3. Logic '0'
4. Logic '1'

The I/O Configuration section allows the system designer to select the source of the GATE input.

COUNTER/TIMER MODES

There are six modes of operation for the counters. In all modes (unless specified otherwise) the following apply:

- GATE, CLK, OUT, CE, and CR apply to any of the three counters as defined previously.
- Counters operate as down counters.
- A clock cycle consist of a rising edge of the counter clock, CLK, followed by a falling edge.
- The rising edge of the clock is used to sample for load or GATE conditions.
- The falling edge of the clock is used to change the value of OUT pin, load the counting element from the count registers (CR ---> CE), or decrement CE (assuming that the proper condition was asserted when sampled on the rising edge of the clock)
- Level sensitive GATE input are sampled on the rising edge of the clock.
- Edge sensitive GATE input set a flip-flop whose output is sampled on the next rising edge of the clock. The flip-flop is reset immediately after it is sampled. This allows detection of consecutive triggers.
- When the control word register (CT_CWR) is written, all control logic is immediately reset , Counter Latch/ReadBack commands canceled, and OUT goes to a known initial state for the selected counter.
- In single byte mode (LSB or MSB), the unused byte of the count value is reset to zero and the counting element (CE) will be loaded with this value.
- In 16-bit count mode (LSB & MSB), if a trigger condition (modes 1 and 5) or a terminal count (modes 2 and 3) occurs between writing the LSB and the MSB, the old count value will be loaded.

Counter/Timer Mode Diagrams

The following keys (shown circled) apply to all modes' diagrams:

Table 11-11.
Counter/Timer Diagrams' Key

KEY	ACTION
1	Write a control word into CT_CWR to operate in the applicable mode as an 8-bit binary counter (LSB)
2	Write an initial count value into CR (via CT_CNTR)
2A	Write a new count value into CR (via CT_CNTR)
3	Load counting element (CE) from CR
4	Decrement CE
5	Freeze CE
6	Rising Edge of GATE (Trigger)

Mode 0: Interrupt on Terminal Count

When the control word register (CT_CWR) is written, the OUT pin for the selected counter goes low. The counting element (CE) is loaded one clock cycle after the counter register (CR) is written (it does not decrement in this cycle). If the counter's GATE is high, the CE will decrement on the next falling edge of the clock (2nd clock cycle after CR is written). If the counter's GATE is low, the CE will not change (OUT remains low) until GATE goes high again and sampled. When the count reaches zero, the OUT pin goes high. It remains high until another count or a new Mode 0 control word is written. The counter keeps decrementing.

Mode 0 can be used for event counting. A count value of N will cause OUT to go high N cycles after the new count is loaded into CE from CR.

If a new count is written into CR, OUT will go low immediately and the new count will be loaded into CE on the next clock cycle. CE will start decrementing on succeeding clock cycles (assuming GATE is high). In the case of a 16-bit count (RW1=RW0=1), the counter is frozen when the LSB is written and OUT will go low immediately. When the MSB is written, CE will be loaded on the next clock cycle. This allows some degree of software synchronization.

If an initial count was written while GATE is low, it will be loaded into CE on the next clock cycle and remain unchanged until GATE goes high. OUT will go high N cycles later.

Writing a new Mode 0 word into CT_CWR will cause CE to stop counting.

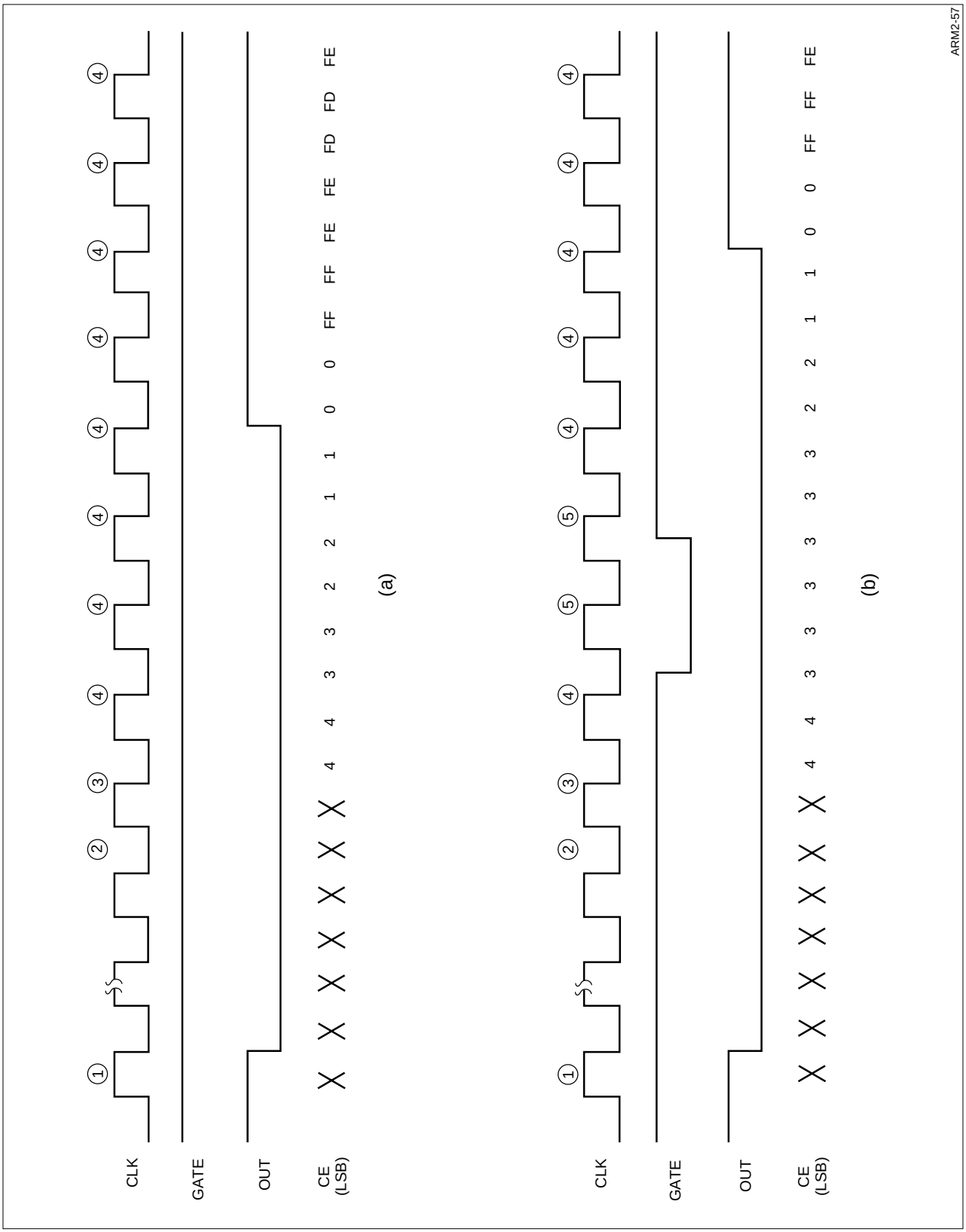


Figure 11-3. Counter/Timer Mode 0: Interrupt On Terminal Count (N = 4)

Mode 1: Hardware Retriggerable One-shot

In this mode, OUT is initially high, writing the control word register (CT_CWR) and the counter register (CR) has no affect on OUT. When there is a rising edge on the GATE input (trigger), counting element (CE) will be loaded from CR on the next clock cycle and OUT goes low. The CE starts decrementing on succeeding clock cycles. When the count reaches zero, the OUT pin goes high. The counter keeps decrementing. If a new trigger is detected, CE is reloaded and the whole process is repeated. If a new trigger is detected in the middle of the count, CE will be reloaded and OUT continues to be LOW until the newly loaded count finishes.

Mode 1 is used to generate a One-Shot. A count value of N will result in a One-Shot pulse (OUT low) of N cycles.

If a new count is written into CR while an existing count is in progress, there will be no effect unless a new trigger is detected. In which case, the new count is loaded into CE and OUT remains low until the new count expires.

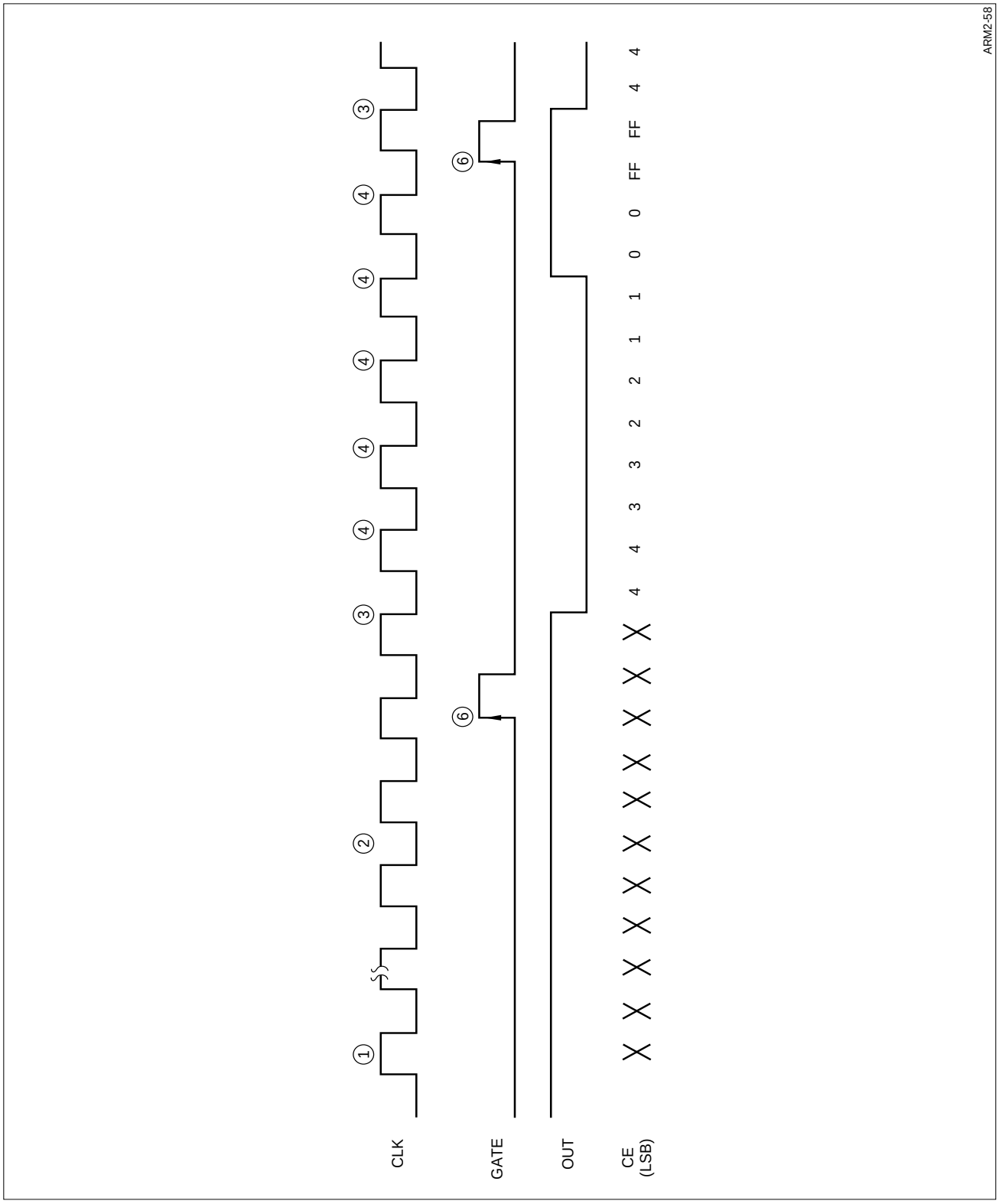


Figure 11-4. Counter/Timer Mode 1: Hardware Retriggerable One-shot (N = 4)

Mode 2: Rate Generator

In this mode, OUT is initially high. When the count value is written to CR, the counting element (CE) is loaded in the next clock cycle. The CE starts decrementing on succeeding clock cycles. The OUT pin will remain high until a count of one is reached, the OUT pin then goes low for one clock cycle. OUT then goes high again, CE is reloaded and the sequence starts again.

If CR is written while CE is counting, it will not affect the current counting sequence. CE will be loaded with the new count at the end of the sequence.

If GATE goes low, OUT goes high immediately and CE stops decrementing. Once GATE goes high again, CE is reloaded on the next clock cycle (regardless of the current count value in CE) and the sequence starts again. The GATE input can be used to synchronize the counter.

Mode 2 functions as a divide-by-N counter. For a count value of N, OUT will be high for N-1 clock cycles and low for 1 clock cycle. The sequence repeats every N clock cycles.

Writing a new Mode 2 word into CT_CWR will cause CE to stop counting.

NOTE: A count value of 1 is illegal in Mode 2

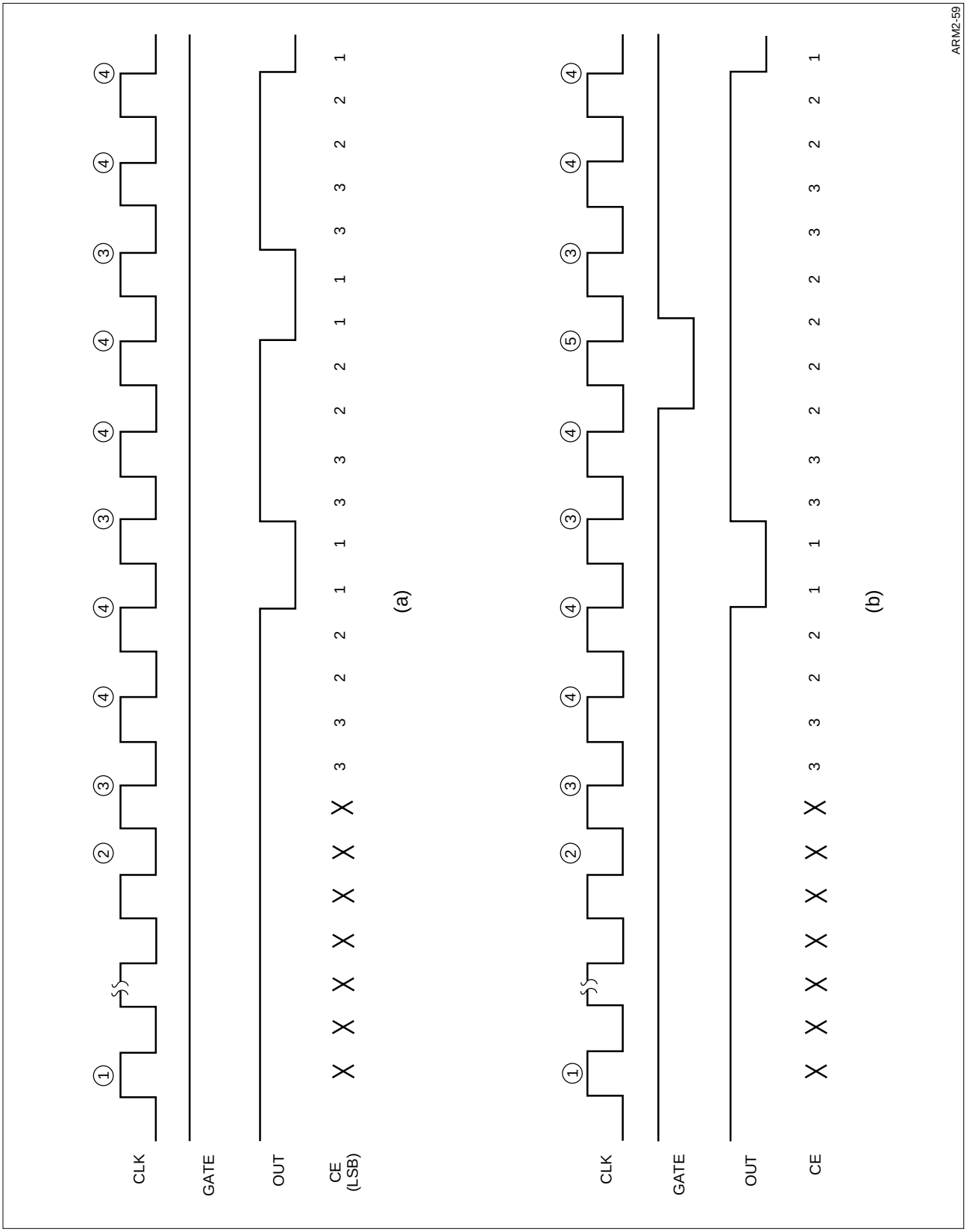


Figure 11-5. Counter/Timer Mode 2: Rate Generator (N = 3)

Mode 3: Square Wave Generator

This mode is similar to Mode 2 except for the duty cycle. In this mode, OUT is initially high. *In this mode, the least significant bit of the count value (even or odd) will always be reset to 0.*

For an initial even count values, when the count value is written to CR, the counting element (CE) is loaded in the next clock cycle. CE is decremented by two on succeeding clock cycles. One clock cycle after CE reaches a count of two, OUT goes low and CE is reloaded. CE is then decremented by two on succeeding clock cycles. One clock cycle after CE reaches a count of two, OUT goes high and CE is reloaded and so on. OUT will be high for N/2 cycles and low for N/2 cycles.

For an initial odd count values, when the count value is written to CR, the counting element (CE) is loaded in the next clock cycle (CE contains the initial odd count -1). CE is decremented by two on succeeding clock cycles. One clock cycle after CE reaches a count of zero, OUT goes low and CE is reloaded. CE is then decremented by two on succeeding clock cycles. One clock cycle after CE reaches a count of two, OUT goes high and CE is reloaded with CR minus one and so on. OUT will be high for (N+1)/2 cycles and low for (N-1)/2 cycles.

If CR is written while CE is counting, it will not affect the current counting sequence. CE will be loaded with the new count when OUT changes level (N/2 for even count or (N+1)/2 for odd count).

If GATE goes low, OUT goes high immediately and CE stops decrementing. Once GATE goes high again, CE is reloaded on the next clock cycle (regardless of the current count value in CE) and the sequence starts again. The GATE input can be used to synchronize the counter.

For a count value of N, a square wave is generated with a period of N clocks.

Writing a new Mode 3 word into CT_CWR will cause CE to stop counting.

NOTE: A count value of 1 is illegal in Mode 3

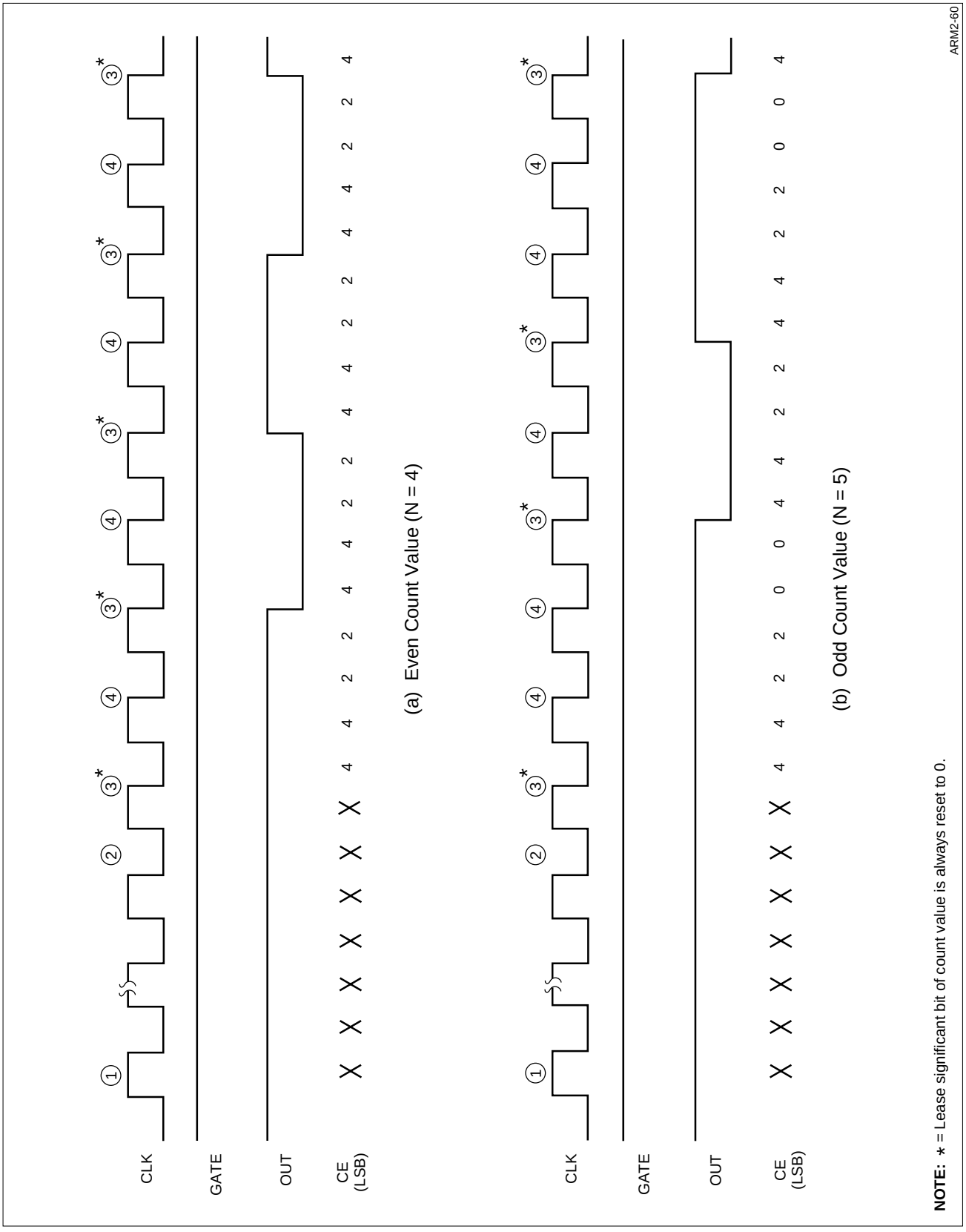


Figure 11-6. Counter/Timer Mode 3: Square Wave Generator

Mode 4: Software Triggered Strobe

In this mode, OUT is initially high. When the count value is written to CR, the counting element (CE) is loaded in the next clock cycle. The CE starts decrementing on succeeding clock cycles. The OUT pin will remain high until a count of zero is reached. The OUT pin then goes low for one clock cycle then goes high again. The counter keeps decrementing.

If CR is written while CE is counting, CE will be loaded with the new count on the next clock cycle and the sequence continues form the new count. When operating as a 16-bit counter (RW1=RW0=1), writing the first byte has no effect, but the second causes the counter to be loaded.

When GATE is high, counting is enabled. When GATE is low, counting is disabled. GATE does not affect OUT.

For a count value of N, OUT will go low N clock cycles after the count is loaded into CE from CR.

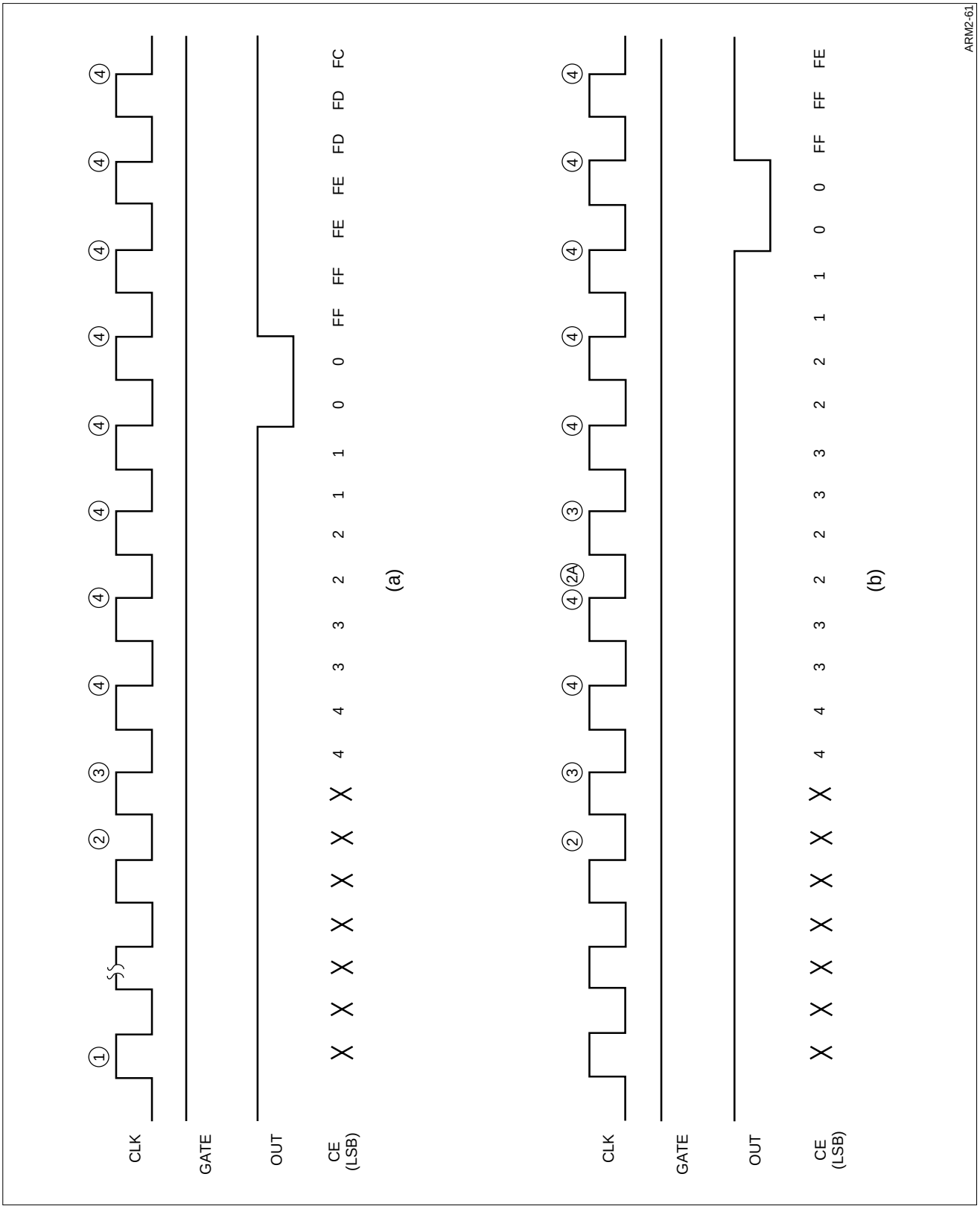


Figure 11-7. Counter/Timer Mode 4: Software Triggered Strobe (N = 4)

Mode 5: Hardware Triggered Strobe

In this mode, OUT is initially high. Once the count value is written to CR, and only after a rising edge on the GATE input (trigger), the counting element (CE) will be loaded from CR on the next clock cycle. The CE starts decrementing on succeeding clock cycles. The OUT pin will remain high until a count of zero is reached. The OUT pin then goes low for one clock cycle then goes high again. The counter keeps decrementing. If a new trigger is detected, CE is reloaded on the next clock cycle and the whole process is repeated.

If GATE re-triggers the counter during a count sequence, then CE is reloaded on the next clock and the sequence starts again.

If CR is written while CE is counting, or at any other time, it will have no effect until the next trigger. When operating as a 16-bit counter (RW1 = RW0 = 1), the second byte must have been written for the trigger to load the new count.

For a count value of N, OUT will go low N+1 clock cycles after the trigger.

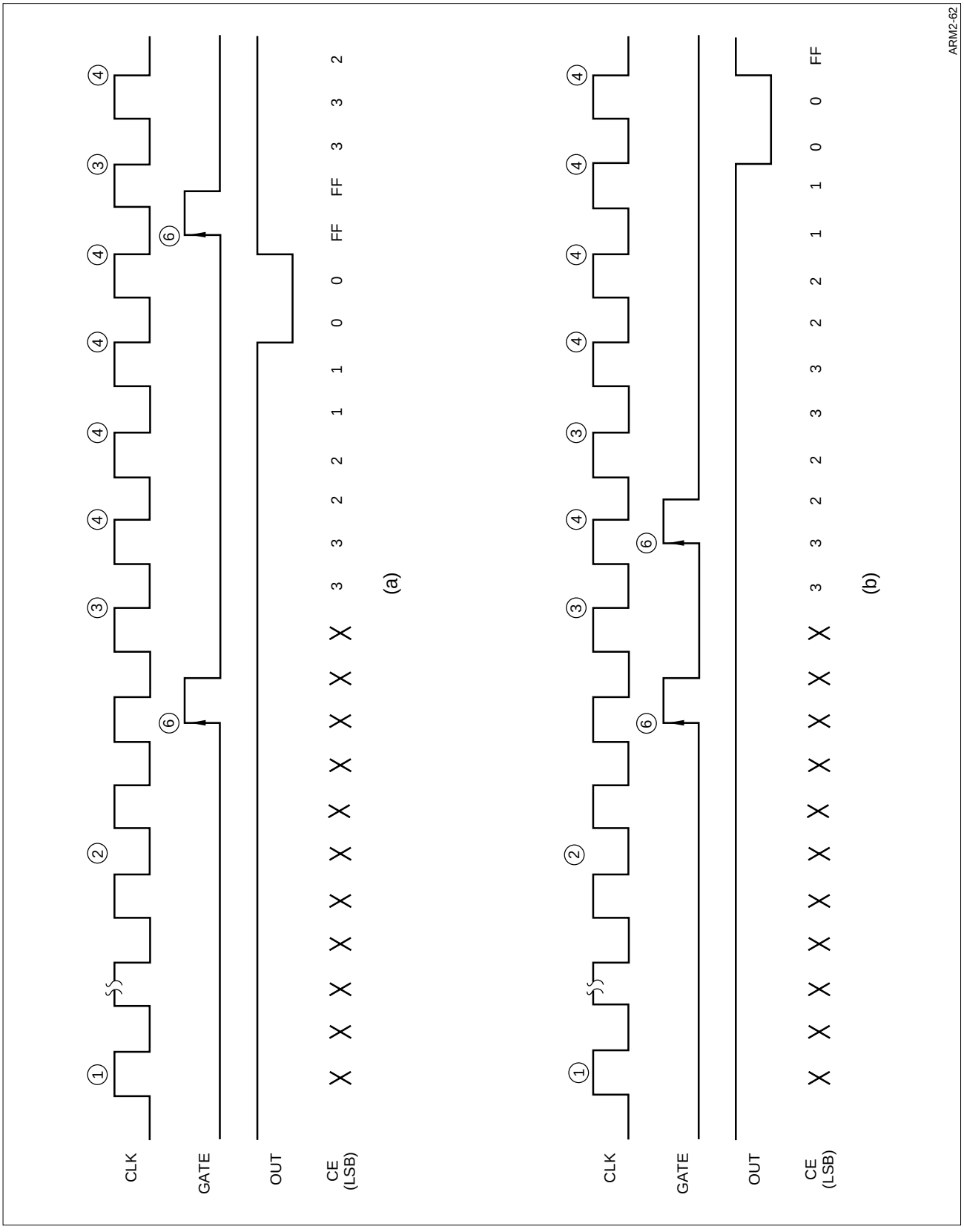


Figure 11-8. Counter/Timer Mode 5: Hardware Triggered Strobe (N = 3)

Mode Summary

The following tables summarize the behavior of the counting elements (CE), and OUT signals.

Table 11-12.
GATE Effect

MODE	*GATE SENSITIVITY*		*GATE STATUS*		
	LEVEL	RISING EDGE	LOW	RISING EDGE	HIGH
0	√		Disable Counting	N/A	Enable Counting
1		√	N/A	On the next clock cycle: (1) Load CE from CR (2) OUT goes low	N/A
2	√	√	(1) Disable Counting (2) Set OUT high	On the next clock cycle, load CE from CR	Enable Counting
3	√	√	(1) Disable Counting (2) Set OUT high	On the next clock cycle, load CE from CR	Enable Counting
4	√		Disable Counting	N/A	Enable Counting
5		√	N/A	On the next clock cycle, load CE from CR	N/A

NOTE: N/A = No Action

Table 11-13.
OUT Summary

MODE	OUT LOW	OUT HIGH	OUT INITIAL STATE
0	(1) New Mode 0 Control word is written into CT_CWR or (2) CR is written	CE = 0	High
1	Rising Edge on GATE	CE = 0	High
2	CE = 1	(1) One Clock Cycle after OUT is Low or (2) GATE is Low	High
3	Even: One Clock Cycle after OUT is High and CE=2 Odd: One Clock Cycle after OUT is High and CE =0	One Clock Cycle after OUT is Low and CE = 2	High
4	CE = 0	One Clock Cycle after OUT is Low	High
5	CE = 0	One Clock Cycle after OUT is Low	High

Table 11-14.
Counter Summary

MODE	CE LOADED	CE FROZEN	CE STARTS COUNTING	CE VALUE AFTER EXPIRATION
0	One Clock Cycle after CR is written	(1) Low GATE or (2) LSB Written to CR in 16-bit Count or (3) New Mode 0 word is Written into CT_CWR	One Clock cycle after CE is Loaded and GATE is High	Binary: 0xFFFF BCD: 9999
1	One Clock Cycle after a Rising Edge on GATE	Keeps Counting	One Clock Cycle after CE is Loaded	Binary: 0xFFFF BCD: 9999
2	One Clock Cycle after: (1) CR is written or (2) CE = 1 or (3) Rising Edge on GATE	(1) GATE is Low or (2) New Mode 2 word is Written into CT_CWR	One Clock Cycle after CE is Loaded and GATE is High	Reload
3	One Clock Cycle after: (1) CR is written or (2) Even: CE = 2 or (3) ODD: OUT is High and CE = 0 or (4) ODD: OUT is Low and CE = 2 or (5) Rising Edge on GATE	(1) GATE is Low or (2) New Mode 3 word is Written into CT_CWR	One Clock Cycle after CE is Loaded and GATE is High	Reload
4	One Clock Cycle after CR is written	GATE is Low	One Clock Cycle after CE is Loaded and GATE is High	Binary: 0xFFFF BCD: 9999
5	One Clock Cycle after a Rising Edge on GATE	Keeps Counting	One Clock Cycle after CE is Loaded	Binary: 0xFFFF BCD: 9999

Table 11-15.
Count Values Limits

MODE	MIN. COUNT	MAX. COUNT*
0	1	0
1	1	0
2	2	0
3	2	0
4	1	0
5	1	0

NOTE: In Binary Mode a count of 0 is equivalent to 2^{16}. In BCD Mode a count of 0 is equivalent to 10^4.

Chapter 12

WATCHDOG TIMER

INTRODUCTION

The Watchdog Timer is a hardware protection against malfunctions. It is a programmable timer that is reset by the software at regular intervals. Failure to do so will cause the 790A to interrupt or reset. As long as the system is operating properly, the software that is executing will clear the timer on a regular basis.

FEATURES

- Eight Programmable Time-Out Intervals
- Freeze Option
- Protection Mechanism
- Selectable Time-Out Action
  - Non-Maskable FIQ Interrupt
  - External Reset
  - System Reset
- Power Management

BLOCK DIAGRAM

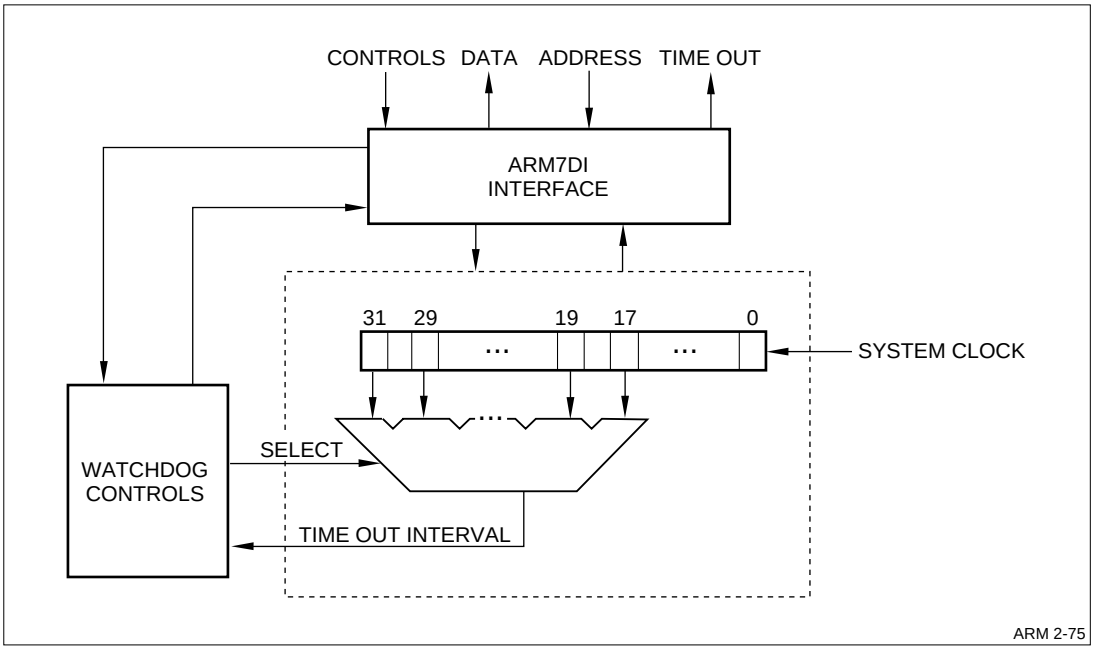


Figure 12-1. Watchdog Timer Block Diagram

REGISTER MAP

The base address for Watchdog Timer register(s) is 0xFFFFAC00. The offset, initial value, access and number of bits for each register is as follows:

Table 12-1.
Register Offset

WATCHDOG REGISTER	ADDRESS OFFSET [7:0]	RESET VALUE	ACCESS	NUMBER OF BITS
WDCTLR	0x30	0x00	R/W	8
WDCNTR	0x34	0x00000000	W	32

GENERAL OPERATION

The Watchdog Timer consists of a 32-bit counter that is used to cause a selectable time-out to protect against malfunctions. The timer needs to be reset by software on a regular basis to prevent a time-out. Failure to do so will result in a time-out that will either cause an interrupt to be taken, an external reset, or a system reset (refer to Figure 12-2).

The Watchdog Timer is controlled through a control register, WDCTLR. There are eight selectable time intervals for time-out: 2^{17}, 2^{19}, 2^{21}, 2^{23}, 2^{25}, 2^{27}, 2^{29}, and 2^{31} clocks for time-outs ranging from 5.24 ms to 1.43 minutes at 25 MHz system clock.

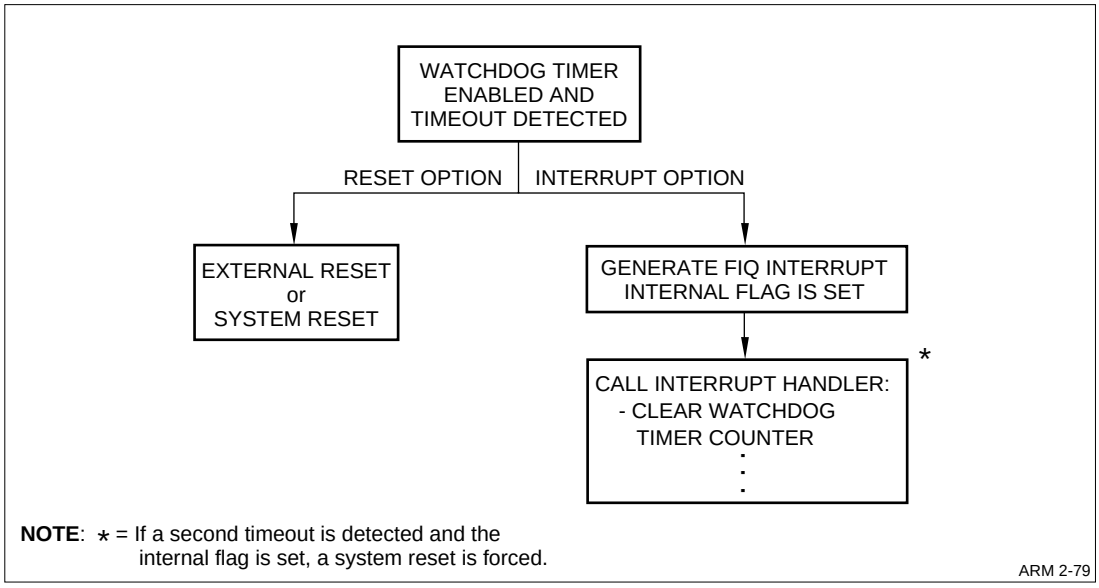


Figure 12-2. Watchdog Timer Action Flow

REGISTER DESCRIPTION

Watchdog Control Register (WDCTLR)

7	6	5	4	3	2	1	0
///	TOP			FRZ	RSP		EN

The Watchdog Timer has an 8-bit control register to control its functionality. Upon reset, the control register is cleared and the Watchdog Timer is disabled.

Table 12-2.
WDCTLR Fields

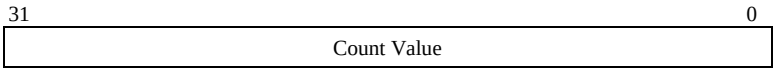
BIT POSITION	NAME	DESCRIPTION
0	EN	Enable/Disable Watchdog: 0 ---> Stop the clock and disable the Watchdog Timer. 1 ---> Enable the clock and Enable the Watchdog Timer. This bit can be locked via the FRZ bit.
2:1	RSP	Time-out Response Selection: 0x ---> Select Non-Maskable FIQ Interrupt. Refer to the Interrupt Controller section. Refer to Protection Mechanism below. 10 ---> Select External Reset. The 790A will drive $\overline{\text{RESETO}}$ LOW for 8 cycles to cause external reset only. The 790A is not affected. 11 ---> Select System Reset. The 790A will reset internally and will drive $\overline{\text{RESETO}}$ LOW for 8 cycles to cause external reset. JTAG TAP Controller is not reset.
3	FRZ	Freeze Enable/Disable: 0 ---> Disable Freeze. 1 ---> Freeze the state of the Enable/Disable bit (EN). This is a SET only flip-flop that once set, it will prevent any writes to the EN bit. It can only be cleared by a system reset. This will prevent the accidental turn on/off of the Watchdog Timer.
6:4	TOP	Time-out Period Select: 000 ---> Select 2^{17} clocks. 001 ---> Select 2^{19} clocks. 010 ---> Select 2^{21} clocks. 011 ---> Select 2^{23} clocks. 100 ---> Select 2^{25} clocks. 101 ---> Select 2^{27} clocks. 110 ---> Select 2^{29} clocks. 111 ---> Select 2^{31} clocks. Refer to the WDCNTR register below for information on 2nd, 3rd, . . . timeouts.
7	///	Reserved

Table 12-3.
Watchdog Timer Time-Out Periods

TOP BIT	TIME-OUT BIT	FIRST TIME-OUT PERIOD*	
		25 MHz	16.7 MHz
000	17	5.24 ms	7.86 ms
001	19	20.97 ms	31.46 ms
010	21	83.89 ms	125.83 ms
011	23	335.54 ms	503.32 ms
100	25	1.34 s	2.01 s
101	27	5.37 s	8.05 s
110	29	21.48 s	32.21 s
111	31	1.43 min	2.15 min

NOTE: *See Counter Register (WDCNTR**)**

Watchdog Counter Register (WDCNTR)



The Watchdog Timer has a free running 32-bit counter to produce a time-out. Writing the counter will clear it regardless of the value on the data bus. Once a time-out period is selected via the control register, a 0 ----> 1 transition on the selected time-out bit (Table 12-3) will cause a time-out event. This time-out event will produce a FIQ interrupt, an external reset, or a system reset as configured in the control register. *The counter keeps on counting and is only cleared when writing it.* The next time-out will occur when the selected time-out bit has another transition from 0 ---> 1. Since the counter keeps on counting after the first time-out, the second, third, ... time-outs will take twice the time to occur (e.g. if the first time-out occurs after 2^{17} cycles, the second, the third, . . . time-out will occur after 2 x 2^{17} cycles unless the counter is cleared).

PROTECTION MECHANISM

The 790A has an added protection in the case where a FIQ interrupt is generated as a result of a time-out. Once a time-out is detected and the Watchdog Timer is programmed to cause an interrupt, an FIQ interrupt is generated and an internal flag is set. If a second time-out is detected and the internal flag is still set, a system reset (internal as well external) is forced. Thus the FIQ interrupt handler must reset this internal flag by clearing the counter (counter is cleared by writing it). This protection mechanism allows recovery in case the interrupt handler is not functioning correctly.

Chapter 13

PROGRAMMABLE PERIPHERAL INTERFACE (PPI)

INTRODUCTION

The 790A provides an 8255-class Programmable Peripheral Interface (PPI). It is a general purpose I/O unit to interface peripherals to the 790A with no external glue logic. It has three 8-bit ports which can be programmed as inputs or outputs. The inputs and outputs can be read or written directly, or they can be strobed by external signals providing handshaking and interrupt signals.

APPLICATIONS

The Programmable Peripheral Interface (PPI) can be used in, but is not limited to:

- Printer Interface
- Keyboard and Display Interface
- D/A and A/D
- Floppy Disk Interface

FEATURES

- 24 Programmable I/O Signals
  - Three 8-bit ports: Port A, Port B, and Port C
- Bit Set/Reset Capability (Port C only)
- Read/Write Control Register
- Three Modes of Operation
  - Basic
  - Strobed
  - Strobed Bi-Directional

BLOCK DIAGRAM

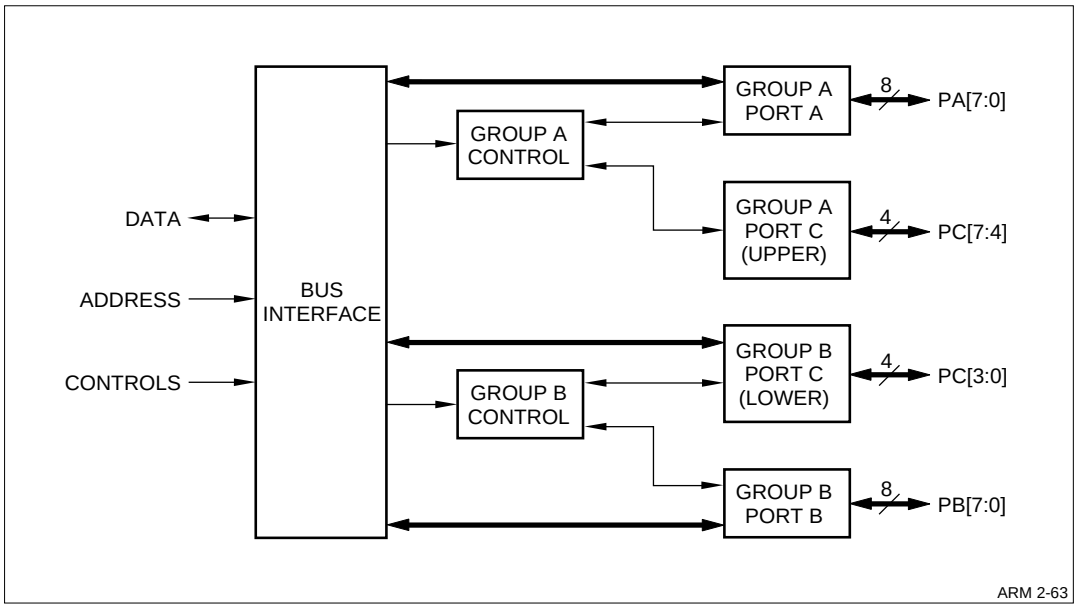


Figure 13-1. Programmable Peripheral Interface (PPI) Block Diagram

EXTERNAL INTERFACE

Table 13-1.
External Interface

SIGNAL	BIT	INPUT/OUTPUT	FUNCTION
PA[7:0]	8	Input/Output	Port A Bus. Can be used as input, output or Bi-Directional data bus.
PB[7:0]	8	Input/Output	Port B Bus. Can be used as input or output data bus.
PC[7:0]	8	Input/Output	Port C Bus. Can be used as input, output data bus, or control/status lines.

REGISTER MAP

The base address for PPI register(s)/Ports is 0xFFFF1C00. The offset, initial value, access and number of bits for each register/Port are as follows:

Table 13-2.
PPI Register Map

REGISTER/ PORT	ADDRESS OFFSET [7:0]	RESET VALUE	ACCESS	NUMBER OF BITS
PPI_PA	0x00	Input/ Mode 0	Read: ARM7DI <--- Port A Write: ARM7DI ---> Port A	8
PPI_PB	0x04	Input/ Mode 0	Read: ARM7DI <--- Port B Write: ARM7DI ---> Port B	8
PPI_PC	0x08	Input/ Mode 0	Read: ARM7DI <--- Port C Write: ARM7DI ---> Port C	8
PPI_CTLR	0x0C	0b0011011	Write: ARM7DI ---> PPI_CTLR	8

UPON RESET

Following chip reset, all ports are in input mode. The PPI Control Register, PPI_CTLR, is initialized to 0b0011011 (All ports are in input mode and Groups A and B are in MODE 0) and INTE flip-flops are cleared. Any port subsequently programmed to become an output port is initialized to LOW level outputs. Port A and Port B modes are defined independently. Port C mode is defined as required by the modes selected for Ports A and B. All output registers (ports), status flip-flops, and INTE flip-flops are cleared when the PPI_CTLR registers is written.

PORTS A, B, AND C

The 24 programmable I/O pins are grouped into three 8-bit ports: PA[7:0], PB[7:0], and PC[7:0]. The three ports form two groups of 12 bits, Group A (PA[7:0] and PC[7:4]) and Group B (PB[7:0] and PC[3:0]).

Port C can be divided into two 4-bit ports via the PPI_CTLR register. When in this mode, Port C signals provide control outputs and status inputs for Ports A and B.

Part of Port B, PB[7:2], and part of Port C, PC[2:0], are multiplexed with UART0 and UART1 modem signals. When the ports are used as modem signals, Group B must be programmed to Mode 0 with PB[7:0] as inputs and PC[2:0] as output. This will allow PB[7:2] to be configured as an input bus to receive the input modem signals into the UARTs and PC[2:0] to be configured as an output bus to drive output modem signals from the UARTs. Refer to the 'I/O Configuration' chapter for more detail.

REGISTER DESCRIPTION

PPI Control Register (PPI_CTLR)

The PPI Control Register, PPI_CTLR, is an 8-bit write-only register. It performs two functions depending on the value of the most significant bit, Bit[7]. If Bit[7] is a logic '1', the register is used to select the operating mode and the data direction of each group. If Bit[7] is a logic '0,' the register is used to Set/Reset the individual bits of Port C.

NOTE: All output registers (ports), status flip-flops, and INTE flip flops are cleared when PPI_CTLR register is written.

Table 13-3.
Bit[7] Functions

BIT [7]	CONTROL REGISTER FUNCTION
0	Bit Set/Reset Port C
1	MODE Selection/Data Direction

Definition of Bits[6:0] in MODE Selection (Bit[7] = 1)

When Bit[7] = 1, the control register bits[6:0] select the operation mode and the data direction for each Group. Bits[2:0] controls Group B. Bits[6:3] controls Group A. The control register bits are shown as follows:

7	6	5	4	3	2	1	0
1	AMH	AML	AI/O	CHI/O	BM	BI/O	CLI/O

Table 13-4
Bit[0]: Group B, Port C (PC[3:0]) Direction

CLI/O	PORT C [3:0] DIRECTION
0	Output
1	Input

Table 13-5
Bit[1]: Group B, Port B Direction

BI/O	PORT B DIRECTION
0	Output
1	Input

Table 13-6.
Bit[2]: Group B MODE Select

BM	GROUP B MODE
0	MODE 0
1	MODE 1

Table 13-7.
Bit[3]: Group A, Port C (PC[7:4]) Direction

CHI/O	PORT C [7:4] DIRECTION
0	Output
1	Input

Table 13-8.
Bit[4]: Group A, Port A Direction

AI/O	PORT A DIRECTION
0	Output
1	Input

Table 13-9.
Bits[6, 5]: Group A Mode Select

AMH	AML	GROUP A MODE
0	0	MODE 0
0	1	MODE 1
1	X	MODE 2

Definition of Bits[6:0] in Set/Reset of Port C (Bit[7] = 0)

When Bit[7] = 0, the control register bits[6:0] are used to Set/Reset the individual bits of Port C. Port C bits can be set or reset by programming the PPI_CTLR Register with Bit[7] = 0.

7	6	5	4	3	2	1	0
0	X	X	X	BS2	BS1	BS0	S/R

Table 13-10.
Bit[0]: Set/Reset Function

S/R	FUNCTION PERFORMED
0	Reset Selected Bit (selected bit = 0)
1	Set Selected Bit (selected bit = 1)

Table 13-11.
Bits[3:1]: Port C Bit Selection

BS_2	BS_1	BS_0	PORT C BIT SELECTED
0	0	0	PC0
0	0	1	PC1
0	1	0	PC2
0	1	1	PC3
1	0	0	PC4
1	0	1	PC5
1	1	0	PC6
1	1	1	PC7

For example, to set PC4, the user needs to write 0x09 to the Control Register. To reset PC2, the user needs to write 0x04 to the Control Register.

INTERRUPT CONTROL

While operating in either MODE 1 or MODE 2, the PPI, via Port C, provides output signals (interrupt request signals) that can generate interrupt signals by externally connecting them to the 790A interrupt pins. The interrupt request signals can be enabled or disabled by setting or resetting the associated INTE flip-flop.

The INTE flip-flops are controlled by the set/reset of different Port C bits depending on the Mode (1 or 2), the Group(A or B) and direction. For example, in MODE 1 with Group A, and Port A as an input, the INTE flip-flop, INTEA2, is controlled by the set/reset of PC4 in the Control Register. Setting PC4 (by writing 0x09 to Control Register) will enable the interrupt (INTEA2 = 1) and resetting it (by writing 0x08 to the Control Register) will disable the interrupt (INTEA2 = 0). All INTE flip-flops are reset during mode changes or chip reset.

Table 13-12.
Interrupt Control

MODE	GROUP	DIRECTION	INTE	INTE CONTROLLED BY SET/RESET OF
1	A	Input	INTEA2	PC4
1	A	Output	INTEA1	PC6
1	B	Input	INTEB	PC2
1	B	Output	INTEB	PC2
2	A	Bi-Directional (Input/Output)	INTEA2/INTEA1	PC4/PC6

NOTE:

1. INTEA2 and INTEA1 represent the same INTE flip-flop for Port A.
2. Mode 0 does not provide interrupts.

OPERATING MODES

MODE Selection

The PPI supports three operational modes as shown below. These modes are controlled by the PPI Control Register, PPI_CTLR, with Bit (7) = 1.

Table 13-13.
PPI Operational Modes

MODE	DESCRIPTION	PPI_CTLR		
		AMH	AML	BM
0	Basic	0	0	0
1	Strobed	0	1	1
2	Strobed Bi-Directional	1	X	X

MODE 0 – Basic

MODE 0 provides for simple Input and Output (no handshaking) for the three ports. Data written to the port is driven off-chip and a read takes data from the port back to the ARM7DI.

MODE 0 consists of two 8-bit ports (Ports A and B) and two 4-bit ports (Port C). Any port can be input or output. Outputs are latched and inputs flow-through without latching. Table 13-14 shows all 16 possible configurations for MODE 0.

Table 13-14.
MODE 0 Port Configurations (Groups A and B in Mode 0)

PPI_CTRLR [7:0]								GROUP A		GROUP B	
BIT [7]	AMH	AML	AI/O	CHI/O	BM	BI/O	CLI/O	PA [7:0]	PC [7:4]	PB [7:0]	PC [3:0]
1	0	0	0	0	0	0	0	Output	Output	Output	Output
1	0	0	0	0	0	0	1	Output	Output	Output	Input
1	0	0	0	0	0	1	0	Output	Output	Input	Output
1	0	0	0	0	0	1	1	Output	Output	Input	Input
1	0	0	0	1	0	0	0	Output	Input	Output	Output
1	0	0	0	1	0	0	1	Output	Input	Output	Input
1	0	0	0	1	0	1	0	Output	Input	Input	Output
1	0	0	0	1	0	1	1	Output	Input	Input	Input
1	0	0	1	0	0	0	0	Input	Output	Output	Output
1	0	0	1	0	0	0	1	Input	Output	Output	Input
1	0	0	1	0	0	1	0	Input	Output	Input	Output
1	0	0	1	0	0	1	1	Input	Output	Input	Input
1	0	0	1	1	0	0	0	Input	Input	Output	Output
1	0	0	1	1	0	0	1	Input	Input	Output	Input
1	0	0	1	1	0	1	0	Input	Input	Input	Output
1	0	0	1	1	0	1	1	Input	Input	Input	Input

Figure 13-2A shows one of the configurations in MODE 0. Figures 13-2B and 13-2C show the timing diagram for MODE 0.

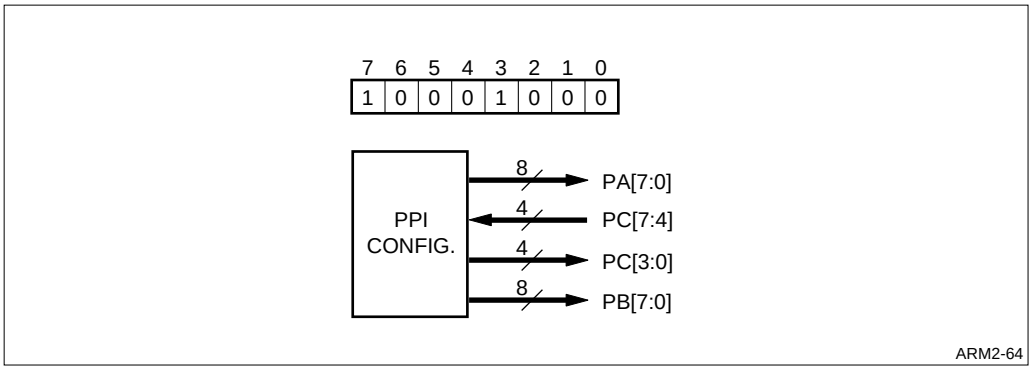


Figure 13-2A. One of MODE0 Configurations

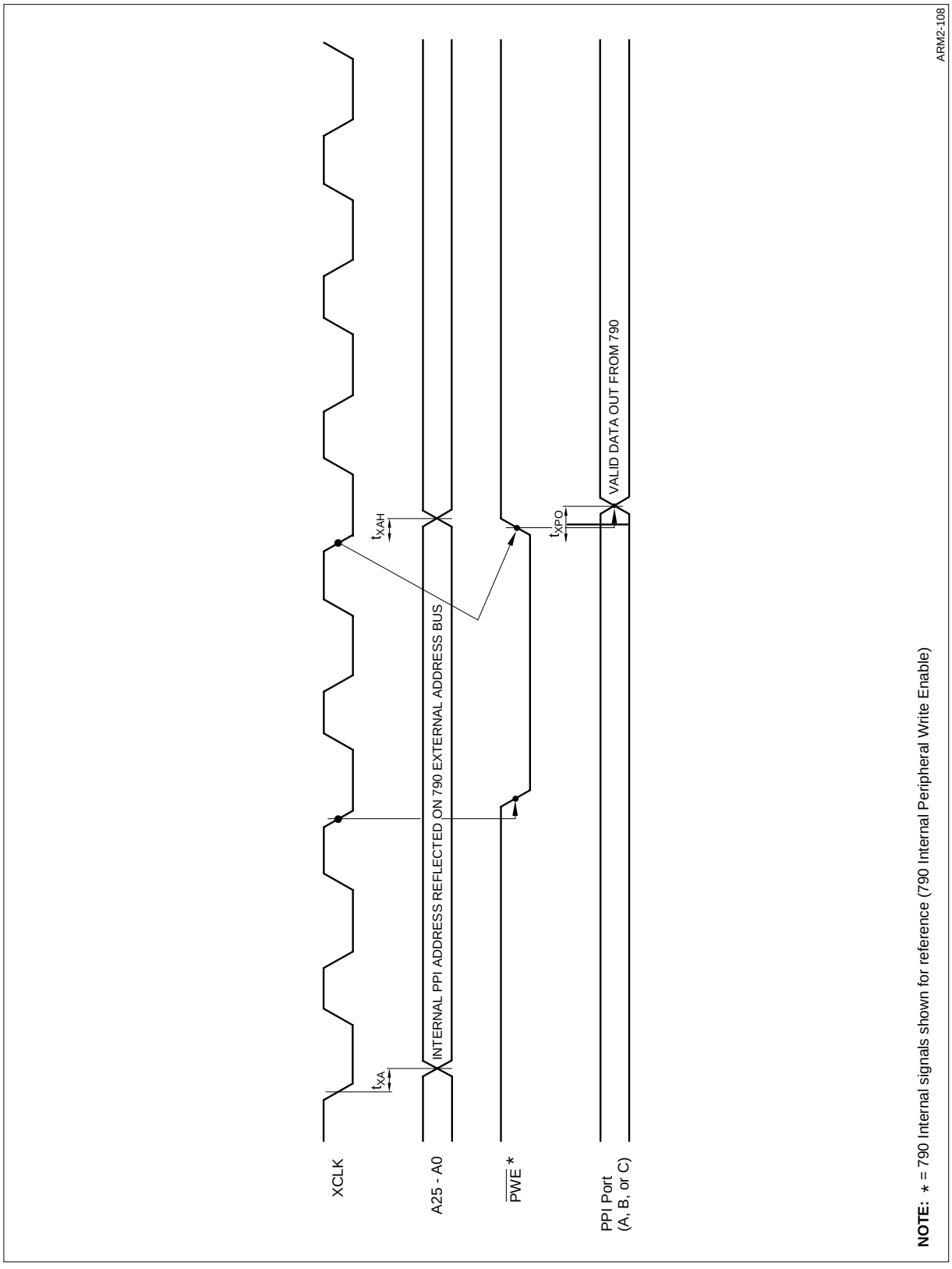


Figure 13-2B. MODE 0 Output Timing

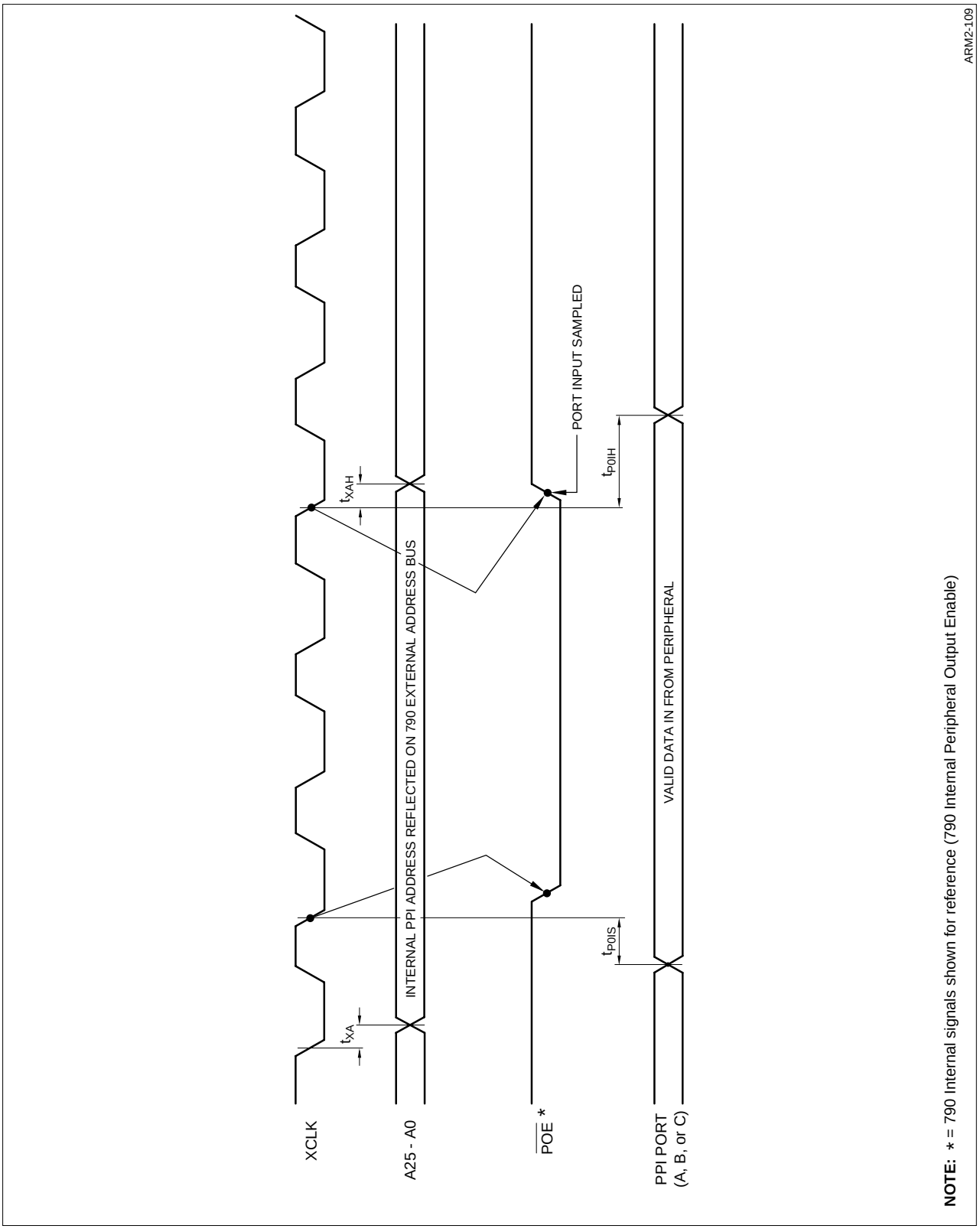


Figure 13-2C. Mode 0, INPUT Timing

MODE 1 (Strobed)

MODE 1 provides for I/O using strobes, or handshaking. In MODE 1, transfers on Ports A and B utilize Port C signals for handshaking.

MODE 1 consists of two groups (Group A and Group B). Each group consists of an 8-bit data port and a 4-bit control/status/data port, Port C. Port A uses PC[7:3] and Port B uses PC[2:0]. The 8-bit data ports are either input or output and both inputs and outputs are latched. Table 13-15 and Figure 13-3a shows all 4 possible configurations for MODE 1. Figure 13-3b shows the timing diagrams for Mode 1, INPUT. Figure 13-3c shows the timing diagrams for MODE 1, OUTPUT.

Table 13-15.
MODE 1 Port Configurations (Groups A and B in Mode 1)

PPI_CTLTR [7:0]								GROUP A	GROUP B
BIT [7]	AMH	AML	AI/O	CHI/O	BM	BI/O	CLI/O		
1	0	1	0	1/0	1	0	X	Output	Output
1	0	1	0	1/0	1	1	X	Output	Input
1	0	1	1	1/0	1	0	X	Input	Output
1	0	1	1	1/0	1	1	X	Input	Input

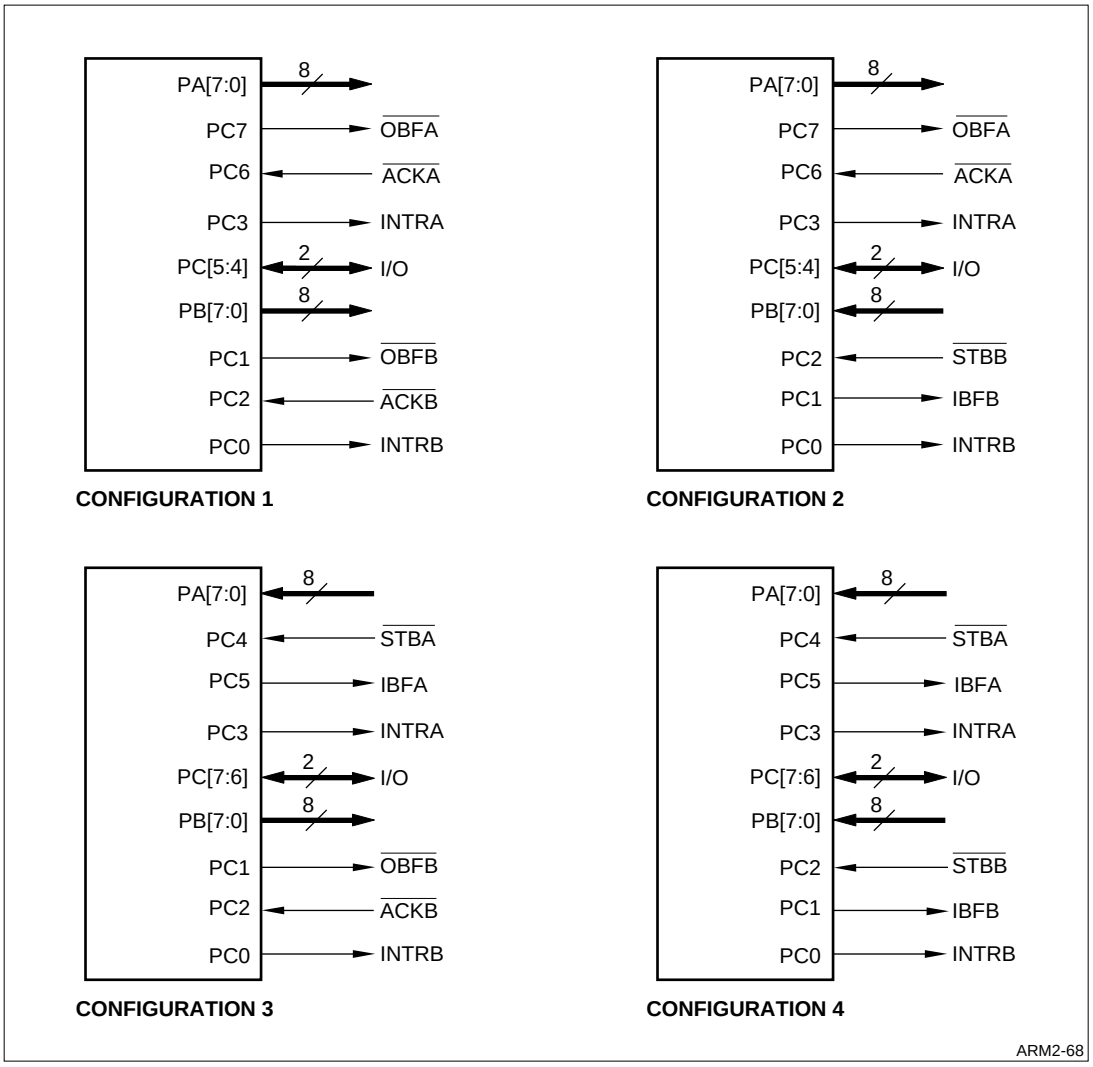


Figure 13-3a. MODE 1 Port Configurations

MODE 1 Input Control Signal Definitions (Group A and Group B)

Table 13-16.
MODE 1 Input Control Signal Definitions

MODE 1 SIGNAL	PORT C EXTERNAL PIN	DIRECTION	DESCRIPTION
$\overline{\text{STBA}}$, $\overline{\text{STBB}}$	PC4, PC2	I	Strobe Input. Active LOW. A LOW level loads data into the input latch causing IBF to be driven HIGH. A LOW to HIGH transition on $\overline{\text{STB}}$ latches the data. The input data must remain valid until after $\overline{\text{STB}}$ goes HIGH.
IBFA, IBFB	PC5, PC1	O	Input Buffer Full. Active HIGH. A HIGH level indicates that data has been loaded into the input latch (acknowledgment). IBF is set when $\overline{\text{STB}}$ goes LOW, and is reset when the ARM7DI reads the data from the port.
INTRA, INTRB	PC3, PC0	O	Interrupt Request. Active HIGH. A HIGH level indicates that the input device is requesting service. INTR is set when $\overline{\text{STB}}$ = IBF = INTE = HIGH. It is cleared when the input data is read by the ARM7DI.
INTEA2	N/A (Internal)	_	Interrupt Enable, Group A. Set to enable interrupts, clear to disable. Controlled by bit Set/Reset to PC4 via the PPI Control Register.
INTEB	N/A (Internal)	_	Interrupt Enable, Group B. Set to enable interrupts, clear to disable. Controlled by bit Set/Reset to PC2 via the PPI Control Register.
PC[7:6]	PC[7:6]	I/O	General Purpose I/O.

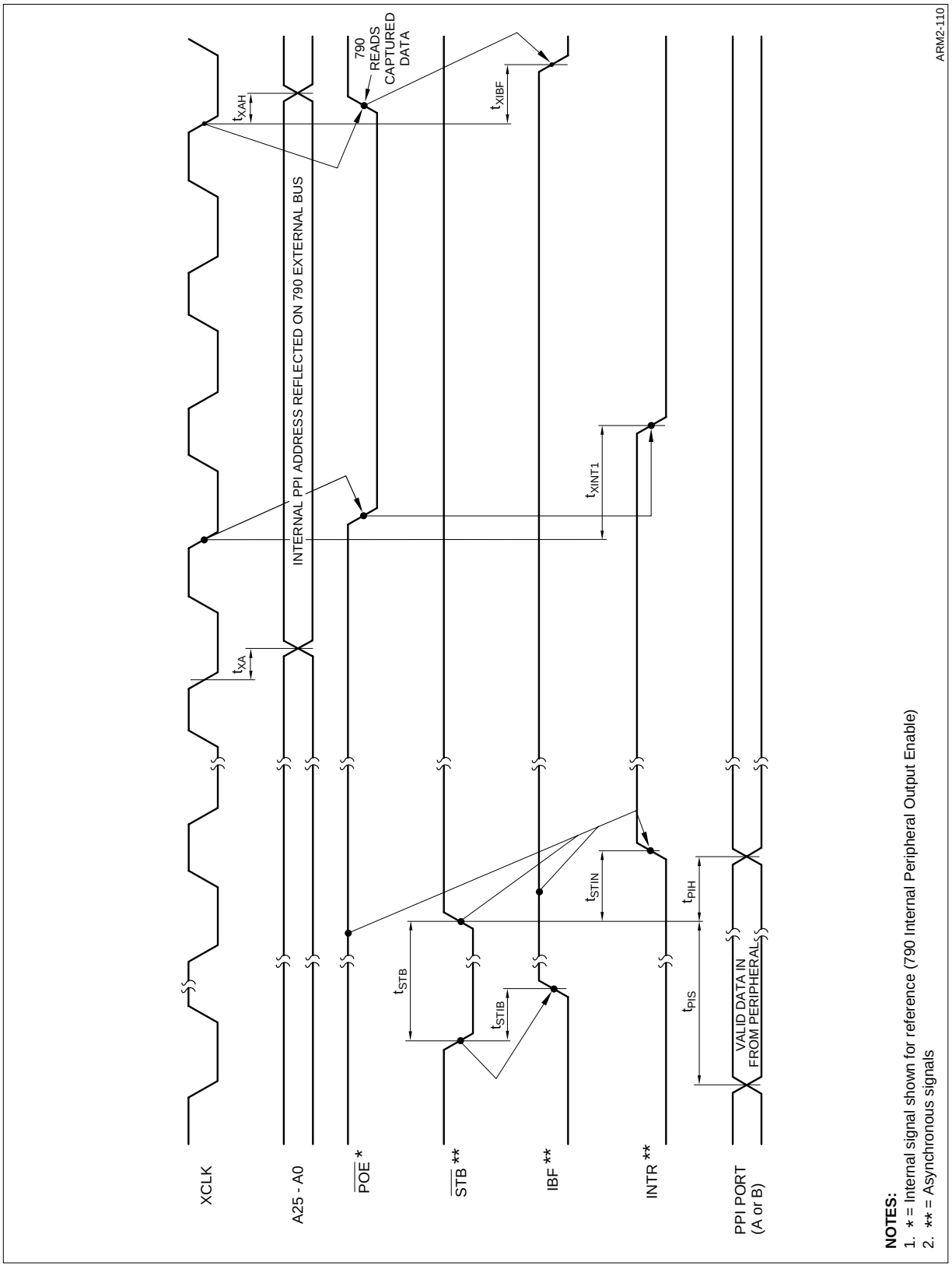


Figure 13-3b. MODE1, INPUT Timing

MODE 1 Output Control Signal Definition (Group A and Group B)

Table 13-17.
MODE 1 Output Control Signal Definition

MODE 1 SIGNAL	PORT C EXTERNAL PIN	DIRECTION	DESCRIPTION
$\overline{\text{ACKA}}$, $\overline{\text{ACKB}}$	PC6, PC2	I	Acknowledge Input. Active LOW. An active LOW level input indicates that data is accepted by the external de-vice, clearing the port's $\overline{\text{OBF}}$ signal.
$\overline{\text{OBFA}}$, $\overline{\text{OBFB}}$	PC7, PC1	O	Output Buffer Full. Active LOW. $\overline{\text{OBF}}$ becomes active when the ARM7DI writes data to the port. $\overline{\text{OBF}}$ becomes inactive when $\overline{\text{ACK}}$ is driven LOW.
INTRA, INTRB	PC3, PC0	O	Interrupt Request. Active HIGH. A HIGH level indicates that the output device has accepted data sent by the ARM7DI. INTR is set when $\overline{\text{ACK}}$ = $\overline{\text{OBF}}$ = INTE = HIGH. It is cleared when the ARM7DI writes data to the port.
INTEA1	N/A (Internal)	-	Interrupt Enable, Group A. Set to enable interrupts, clear to disable. Controlled by bit Set/Reset to PC6 via the PPI Control Register.
INTEB	N/A (Internal)	-	Interrupt Enable, Group B. Set to enable interrupts, clear to disable. Controlled by bit Set/Reset to PC2 via the PPI Control Register.
PC[5:4]	PC[5:4]	I/O	General Purpose I/O.

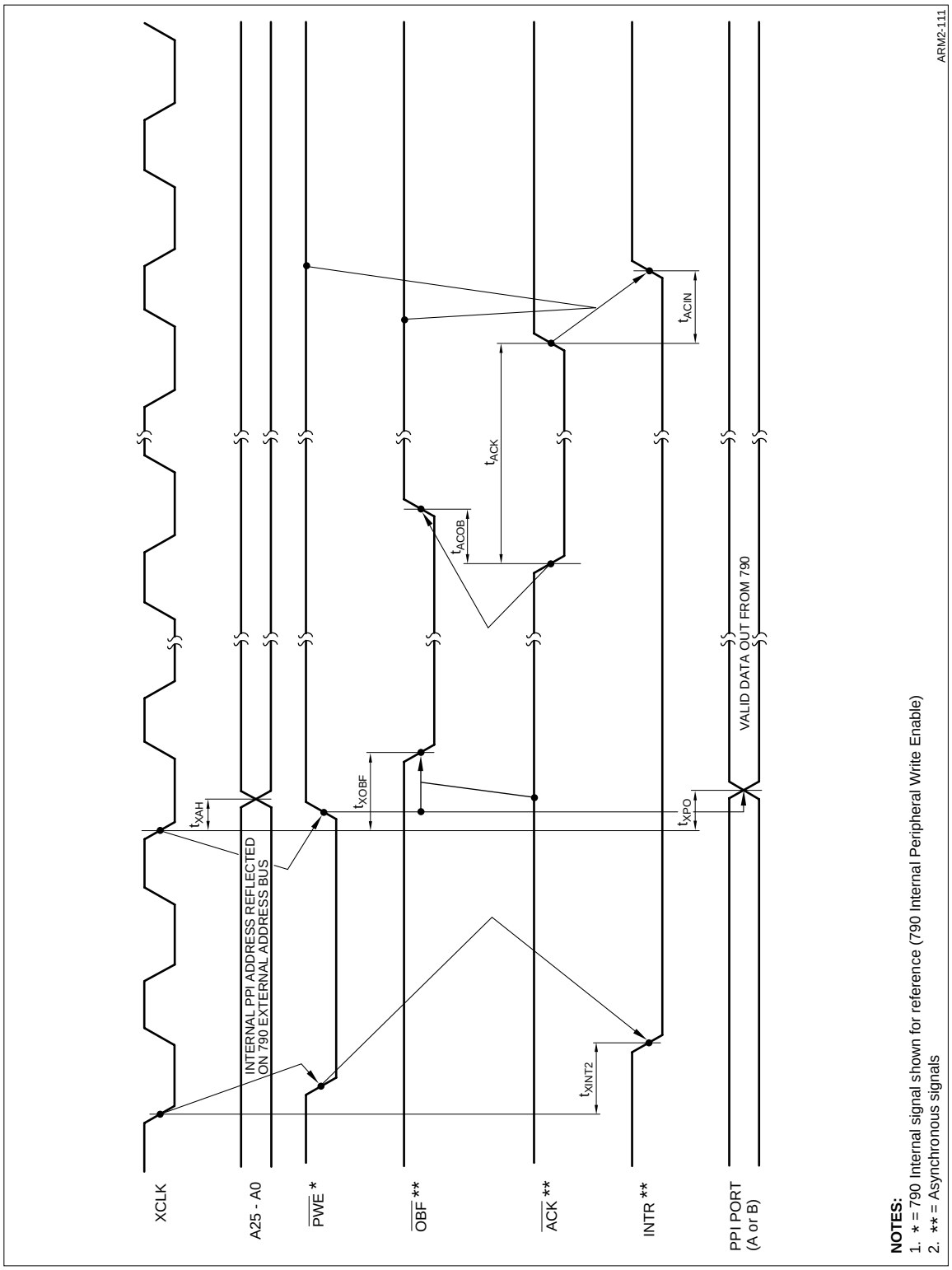


Figure 13-3c. MODE1, OUTPUT Timing

MODE 2 – Strobed Bi-Directional (Port A Only)

MODE 2 operation on Port A provides for Bi-Directional data transfer on the 8-bit port with handshaking on Port C as in MODE 1. Group B is not used in this mode but can be programmed to operate in MODEs 0 or 1.

MODE 2 consists of an 8-bit Bi-Directional bus (Port A) and 5-bit control bits (Port C) to control Port A. Both Inputs and Outputs are latched. Group B can be programmed to MODE 0 or MODE 1. Table 13-18 and Figure 13-4a shows all 4 possible configurations for MODE 2.

Table 13-18.
MODE 2 Port Configurations

PPI_CTLTR [7:0]								GROUP A	GROUP B
BIT [7]	AMH	AML	AI/O	CHI/O	BM	BI/O	CLI/O		
1	1	X	X	X	0	0	1/0	Bi-Directional	Mode 0/Output
1	1	X	X	X	0	1	1/0	Bi-Directional	Mode 0/Input
1	1	X	X	X	1	0	X	Bi-Directional	Mode 1/Output
1	1	X	X	X	1	1	X	Bi-Directional	Mode 1/Input

NOTE: X = Don't Care

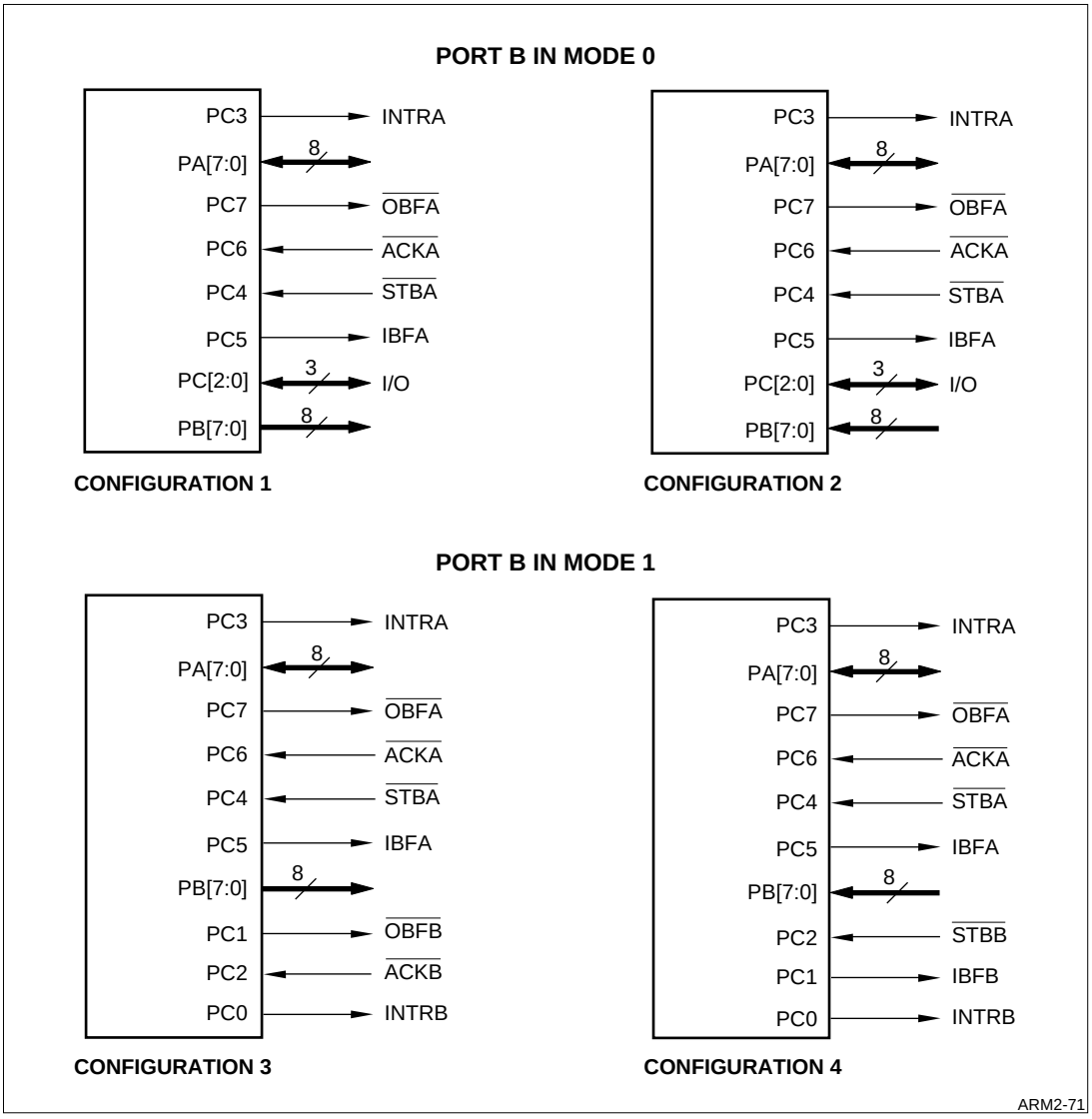


Figure 13-4a. MODE2 Port Configurations

MODE 2 Bidirectional Control Signal Definition

Table 13-19.
MODE 2 Bidirectional Control Signal Definition

MODE 2 SIGNAL	PORT C EXTERNAL PIN	DIRECTION	DESCRIPTION
INTRA	PC3	O	Interrupt Request. Active HIGH. An active HIGH level can interrupt the ARM7DI for either input or output transfers. For INPUT transfers, the interrupt is set when $\overline{STBA}$ = IBFA = INTEA2 = HIGH and cleared when the input data is read by the ARM7DI. For OUTPUT transfers, the interrupt is set when $\overline{ACKA}$ = $\overline{OBFA}$ = INTEA1 = HIGH and cleared when the ARM7DI writes data to the port.
$\overline{OBFA}$	PC7	O	Output Buffer Full. Active LOW. Becomes active LOW when the ARM7DI write data to Port A. $\overline{OBFA}$ becomes in-active when $\overline{ACKA}$ goes LOW.
$\overline{ACKA}$	PC6	I	Acknowledge. Active LOW. An active LOW level allows the data in Port A latch to be sent out on the Port A bus. When $\overline{ACKA}$ is HIGH, Port A is in high impedance. A low level on $\overline{ACKA}$ clears $\overline{OBFA}$.
INTEA1	N/A (Internal)	_	Interrupt Enable 1 for Port A. INTE1 is the Interrupt Enable register associated with output data transfers ($\overline{OBFA}$). INTE1 is controlled by bit set/reset operations on bit PC6 via the PPI Control Register.
$\overline{STBA}$	PC4	I	Strobe Input. Active LOW. An active LOW level loads input data into the Port A latch, causing IBFA to be driven HIGH. A LOW to HIGH transition on $\overline{STB}$ latches the data. The input data must remain valid until after $\overline{STBA}$ goes HIGH.
IBFA	PC5	O	Input Buffer Full. Active HIGH. A HIGH level indicates that the data has been loaded into Port A latch. IBFA is set when $\overline{STBA}$ goes LOW, and reset when the ARM7DI reads the data from Port A.
INEA2	N/A (Internal)	_	Interrupt Enable 2 for Port A. INTE2 is the Interrupt Enable register associated with input data transfers (IBFA). INTE2 is controlled by bit set/reset operations on bit PC4 via the PPI Control Register

PC[0:2] can function as a general-purpose I/O if Group B is in MODE 0, and control/status lines if Group B is in MODE 1. Figure 13-4B shows the timing diagram for MODE 2.

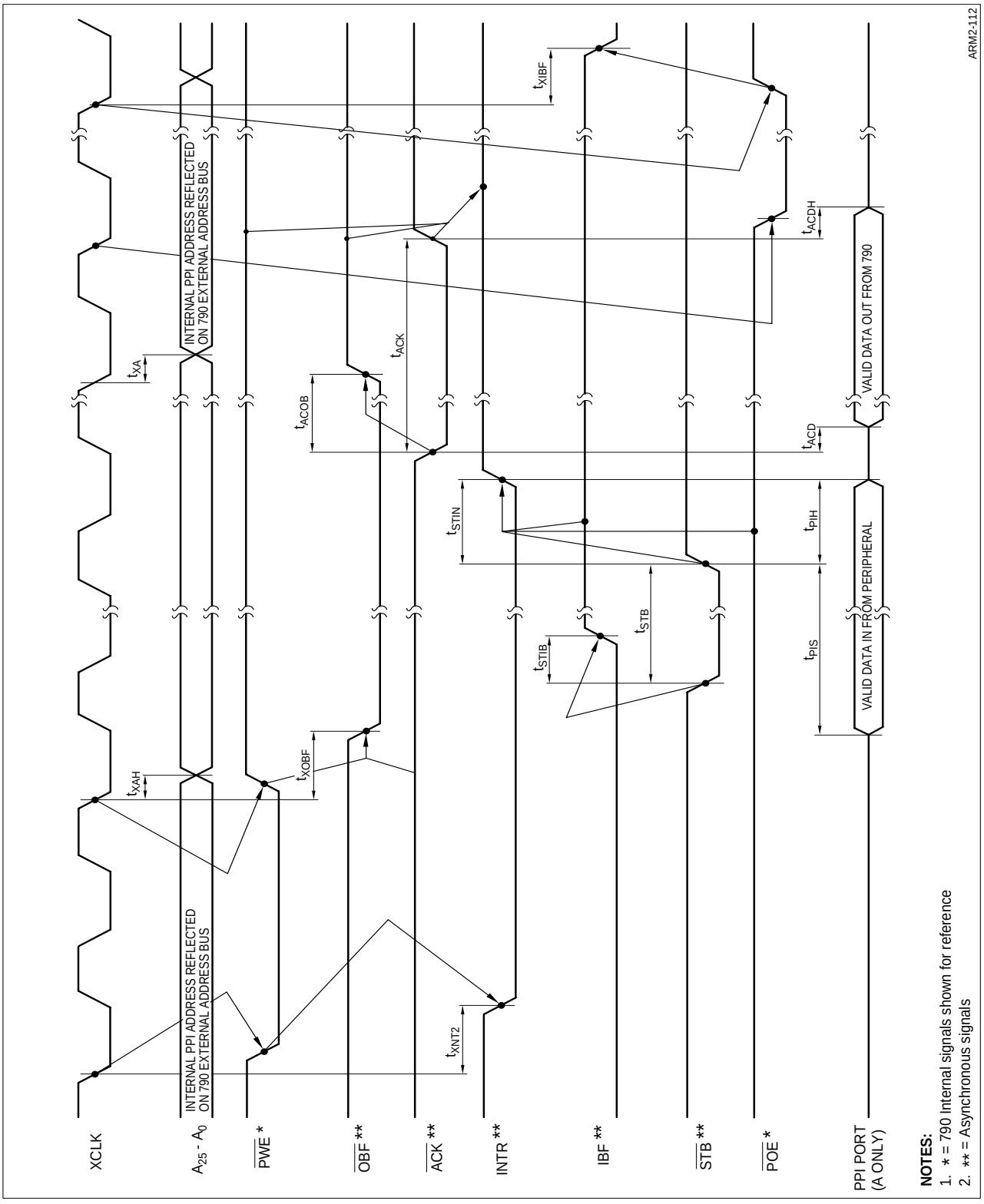


Figure 13-4b. MODE 2, Bi-Directional Timing

MODE Definition Summary

The following tables summarize all possible combinations for each MODE.

Table 13-20.
MODE 0: Groups A and B are both in Mode 0

PA [7:0]	PC[7:4]	PB[7:0]	PC[3:0]
Output	Output	Output	Output
Output	Output	Output	Input
Output	Output	Input	Output
Output	Output	Input	Input
Output	Input	Output	Output
Output	Input	Output	Input
Output	Input	Input	Output
Output	Input	Input	Input
Input	Output	Output	Output
Input	Output	Output	Input
Input	Output	Input	Output
Input	Output	Input	Input
Input	Input	Output	Output
Input	Input	Output	Input
Input	Input	Input	Output
Input	Input	Input	Input

Table 13-21.
MODE 1: Groups A and B are both in Mode 1

PORT	CONFIGURATION 1	CONFIGURATION 2	CONFIGURATION 3	CONFIGURATION 4
PA[7:0]	Output	Output	Input	Input
PB[7:0]	Output	Input	Output	Input
PC0	INTRB	INTRB	INTRB	INTRB
PC1	$\overline{\text{OBFB}}$	IBFB	$\overline{\text{OBFB}}$	IBFB
PC2	$\overline{\text{ACKB}}$	$\overline{\text{STBB}}$	$\overline{\text{ACKB}}$	$\overline{\text{STBB}}$
PC3	INTRA	INTRA	INTRA	INTRA
PC4	I/O	I/O	$\overline{\text{STBA}}$	$\overline{\text{STBA}}$
PC5	I/O	I/O	$\overline{\text{IBFA}}$	$\overline{\text{IBFA}}$
PC6	$\overline{\text{ACKA}}$	$\overline{\text{ACKA}}$	I/O	I/O
PC7	$\overline{\text{OBFA}}$	$\overline{\text{OBFA}}$	I/O	I/O

Table 13-22.
MODE 2

PORT	CONFIGURATION 1	CONFIGURATION 2	CONFIGURATION 3	CONFIGURATION 4
PA[7:0]	BiDirectional	BiDirectional	BiDirectional	BiDirectional
PB[7:0]	MODE 0 (Out)	MODE 0 (In)	MODE 1 (Out)	MODE 1 (In)
PC0	I/O	I/O	INTRB	INTRB
PC1	I/O	I/O	$\overline{\text{OBFB}}$	IBFB
PC2	I/O	I/O	$\overline{\text{ACKB}}$	$\overline{\text{STBB}}$
PC3	INTRA	INTRA	INTRA	INTRA
PC4	$\overline{\text{STBA}}$	$\overline{\text{STBA}}$	$\overline{\text{STBA}}$	$\overline{\text{STBA}}$
PC5	$\overline{\text{IBFA}}$	$\overline{\text{IBFA}}$	$\overline{\text{IBFA}}$	$\overline{\text{IBFA}}$
PC6	$\overline{\text{ACKA}}$	$\overline{\text{ACKA}}$	$\overline{\text{ACKA}}$	$\overline{\text{ACKA}}$
PC7	$\overline{\text{OBFA}}$	$\overline{\text{OBFA}}$	$\overline{\text{OBFA}}$	$\overline{\text{OBFA}}$

Table 13-23.
Valid Modes

GROUP A	GROUP B
Mode 0	Mode 0
Mode 0	Mode 1
Mode 1	Mode 0
Mode 1	Mode 1
Mode 2	Mode 0
Mode 2	Mode 1

PORT WRITES

All ports can be written to, regardless of the mode of operation of the device. When operating in input mode, the byte written is kept in the latch but not used. When operating in output mode, the byte written can be read back without affecting the state of the output.

READING/WRITING PORT C

Port C is a multi-purpose port. Its bits are used for data (I/O), control, and status depending on the mode of operation. To read Port C, the user must use a normal read operation (similar to Ports A and B). The data read depends on the mode of operation. To write Port C, there are two ways of doing so, a normal write operation (similar to Ports A and B) or a Set/Reset operation using the Control Register. The following sections will explain the details of reading/writing of Port C.

MODE 0

Port C Function:

- Can be either an input, output, or a combination of both.
- Refer to MODE 0 Definition Summary in Table 13-20.

Read Operation:

- A normal read operation (PPI_PC).
- Returns the value of all bits on Port C.

Write Operation (Normal):

- A normal write operation (PPI_PC).
- Only output bits are written.
- Input bits are not affected.
- Interrupt Enable (INTEA1, INTEA2, INTEB) bits are not affected.

Write Operation (Set/Reset):

- Program P[3:0] and/or PC[7:4] as output.
- Use PPI_CTLR with Bit[7] = 0 to Set/Reset individual bits of Port C.

MODE 1

Port C Function:

- Control, status, and I/O.
- Refer to MODE 1 Definition Summary in Table 13-21.

Read Operation:

- A normal read operation.
- INTE bits replace the status input bits ($\overline{\text{STB}}$ and $\overline{\text{ACK}}$).
- Table 13-24 shows the result of the read operation is a function of the configuration in Table 13-21.

Table 13-24.
MODE 1 Read Operation Configuration

CONFIGURATION NUMBER	D7	D6	D5	D4	D3	D2	D1	D0
1	$\overline{\text{OBFA}}$	INTEA1*	I/O	I/O	INTRA	INTEB*	$\overline{\text{OBFB}}$	INTRB
2	$\overline{\text{OBFA}}$	INTEA1*	I/O	I/O	INTRA	INTEB*	IBFB	INTRB
3	I/O	I/O	IBFA	INTEA2*	INTRA	INTEB*	$\overline{\text{OBFB}}$	INTRB
4	I/O	I/O	IBFA	INTEA2*	INTRA	INTEB*	IBFB	INTRB

NOTE: INTE bits replace the status input bits ($\overline{\text{STB}}$ and $\overline{\text{ACK}}$) when reading port C in MODE 1.

Write Operation:

- Only Set/Reset operation to write all Port C output bits (including I/O, IBF, $\overline{OBF}$, INTR). This is done by using PPI_CTLR with Bit[7] = 0.
- Port C general purpose input bits (data) are not affected.
- Port C Status input bits ($\overline{ACK}$ or $\overline{STB}$) are not affected, instead the appropriate INTE bit is set/reset as defined in the Interrupt Control section.

MODE 2

Port C Function:

- Control, status, and I/O.
- Refer to MODE 2 definition summary in Table 13-22.

Read Operation:

- A normal read operation.
- INTE bits replace the status input bits ($\overline{STB}$ and $\overline{ACK}$) .
- The result of the read operation is a function of the configuration as shown in Table 13-25.

Table 13-25.
MODE 2 Read Operation Configuration

CONFIGURATION NUMBER	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	$\overline{OBFA}$	INTEA1*	IBFA	INTEA2*	INTRA	I/O	I/O	I/O
2	$\overline{OBFA}$	INTEA1*	IBFA	INTEA2*	INTRA	I/O	I/O	I/O
B3	$\overline{OBFA}$	INTEA1*	IBFA	INTEA2*	INTRA	INTEB*	$\overline{OBFA}$	INTRB
4	$\overline{OBFA}$	INTEA1*	IBFA	INTEA2*	INTRA	INTEB*	IBFB	INTRB

NOTE: INTE bits replace the status input bits ($\overline{STB}$ and $\overline{ACK}$) when reading Port C in MODE 2.

Write Operation:

- Only Set/Reset operation to write all Port C output bits (including I/O, IBF, $\overline{OBF}$, INTR).
- Port C general purpose input bits (data) are not affected.
- Port C Status input bits ($\overline{ACK}$ or $\overline{STB}$) are not affected, instead the appropriate INTE bit will be set/reset as defined in the Interrupt Control section.

Chapter 14

INTERRUPT CONTROLLER

INTRODUCTION

The 790A Interrupt Controller supports both internal and external interrupt sources. Internally, there are seven peripheral interrupt sources (3 UARTs, 3 Counters/Timers, and a Watchdog Timer). Externally there are six interrupt sources (INT[5:0]).

FEATURES

- Six External Interrupts
- Seven Internal Interrupts
- Low Interrupt Latency
- Enable/Disable
- Active High or Low
- Drive $\overline{\text{IRQ}}$ or $\overline{\text{FIQ}}$
- Level or Edge Triggered
- ARM7DI Wake Up

BLOCK DIAGRAM

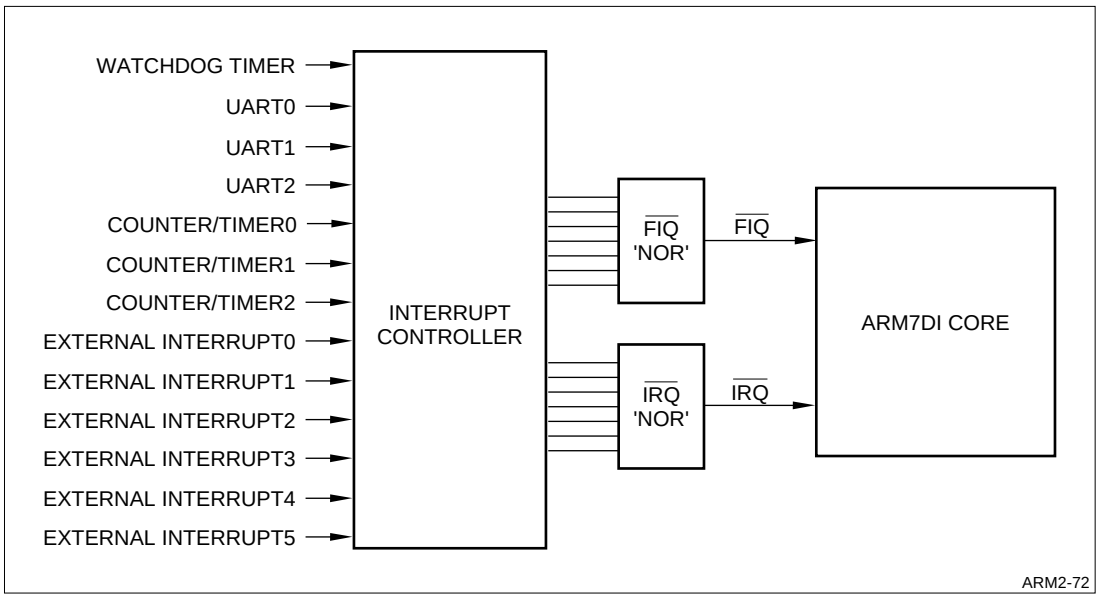


Figure 14-1. Interrupt Controller Block Diagram

INTERNAL DIAGRAM

Figure 14-2 details the interrupt flow for a fully programmable interrupt channel (e.g. channels 0 - 5). Channels 6 - 12 have a similar flow.

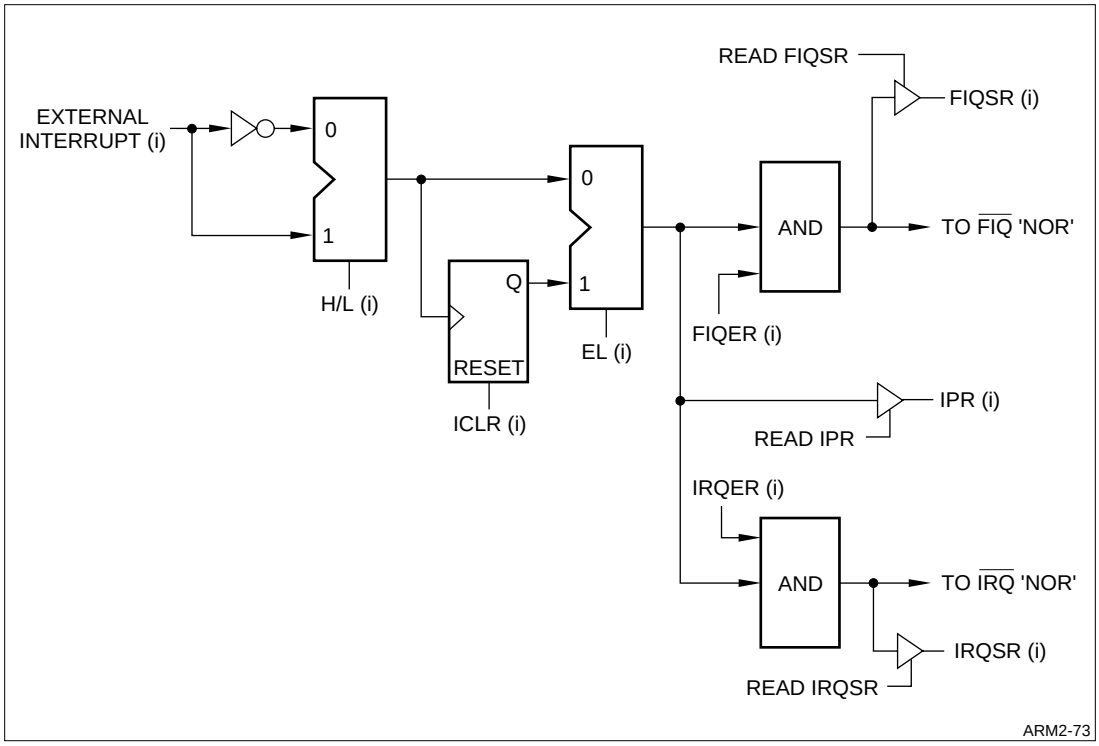


Figure 14-2. Channel (i) Interrupt Flow

REGISTER MAP

The base address for the Interrupt Controller registers is 0xFFFFA800. The offset, initial value, access and number of bits for each register are as follows.

Table 14-1.
Interrupt Controller Register Map

REGISTER	ADDRESS OFFSET [7:0]	RESET VALUE	ACCESS	NUMBER OF BITS
ICR0	00	0x00	R/W	16
ICR1	04	0x00	R/W	8
ICLR	08	0x00	W	16
IRQER	0C	0x00	R/W	16
FIQER	10	0x00	R/W	16
IRQSR	14	0x00	R	16
FIQSR	18	0x00	R	16
IPR	1C	0x00	R	16

INTERRUPT CHANNEL ASSIGNMENT

Table 14-2.
Interrupt Channel Assignment

DEVICE	INTERRUPT CHANNEL	IRQ/FIQ	LEVEL/EDGE TRIGGER	LOW/HIGH ACTIVE
External Interrupt0 (INT0)	0	Programmable	Programmable	Programmable
External Interrupt1 (INT1)	1	Programmable	Programmable	Programmable
External Interrupt2 (INT2)	2	Programmable	Programmable	Programmable
External Interrupt3 (INT3)	3	Programmable	Programmable	Programmable
External Interrupt4 (INT4)	4	Programmable	Programmable	Programmable
External Interrupt5 (INT5)	5	Programmable	Programmable	Programmable
Counter/Timer0	6	Programmable	Edge	Programmable
Counter/Timer1	7	Programmable	Edge	Programmable
Counter/Timer2	8	Programmable	Edge	Programmable
UART0	9	Programmable	Level	High
UART1	10	Programmable	Level	High
UART2	11	Programmable	Level	High
Watchdog Timer	12	FIQ only	Edge	–

GENERAL OPERATION

The Interrupt Controller for the 790A manages both internal and external interrupts. It provides the ability to configure, enable, clear, provide status, and poll interrupts.

Interrupts can be enabled as an IRQ or FIQ. The ARM7DI has two types of interrupts, a fast high priority interrupt ($\overline{FIQ}$) and a low priority interrupt ($\overline{IRQ}$). The Interrupt Controller will steer the proper interrupts to either FIQ or IRQ according to its configuration. All IRQ interrupts are combined together to obtain the final IRQ to the ARM7DI and all FIQ interrupts are combined together to obtain the final FIQ. The ARM7DI samples interrupts on the falling edge of the system clock, XCLK. All interrupts are synchronized internal to the ARM7DI. The ARM7DI checks for active interrupts (after synchronization) at the end of each instruction. Refer to the *ARM7DI Data Sheet* for additional information.

External interrupts can be configured as level or edge triggered. Edge triggered interrupts will be latched to preserve its value prior to processing. Level triggered interrupts will not be latched and must remain active long enough for the 790A to detect and process.

Current interrupt status (masked by the enable registers, IRQER/FIQER) is indicated in the IRQ and FIQ status registers, IRQSR/FIQSR. These two registers allow the interrupt handler to examine the appropriate status (IRQ or FIQ) in one read operation thus reducing interrupt latency.

The polling register, IPR, is used to examine the interrupt status prior to masking. The clear register, ICLR, is used to clear edge triggered interrupts only. Level triggered interrupts should be cleared at the interrupting device.

REGISTER DESCRIPTION

Interrupt Configuration Registers (ICR0/ICR1)

ICR0

7	6	5	4	3	2	1	0
CH3		CH2		CH1		CH0	
E/L_3	H/L_3	E/L_2	H/L_2	E/L_1	H/L_1	E/L_0	H/L_0

15	14	13	12	11	10	9	8
				CH5		CH4	
///	///	///	///	E/L_5	H/L_5	E/L_4	H/L_4

ICR1

7	6	5	4	3	2	1	0
					CH8	CH7	CH6
///	///	///	///	///	H/L_2	H/L_1	H/L_0

Table 14-3.
ICR0/ICR1 Fields

FIELD NAME	DESCRIPTION
E/L_i	0: Level-Triggered 1: Edge-Triggered
H/L_i	0: Active Low (Level Triggered) or High ---> Low (Edge Triggered) 1: Active High (Level Triggered) or Low ---> High (Edge Triggered)
///	Reserved

Register ICR0 is used to configure the external interrupts. Each external interrupt can be configured to level or edge triggered and low or high active. Register ICR1 is used to configure the Counter/Timer interrupts for low or high active. Counter/Timer interrupts are always edge triggered interrupts. UART interrupts are always level triggered and high active. Watchdog timer is always edge triggered. IRQ and FIQ configuration is done via IRQER and FIQER enable registers.

Interrupt Clear Register (ICLR)

7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

15	14	13	12	11	10	9	8
///	///	///	///	///	///	CH12	CH8

Table 14-4.
ICLR Fields

FIELD NAME	DESCRIPTION
CHi	0: No Action 1: Clear Channel i Edge Level Interrupt
///	Reserved

The Interrupt Clear Register is used to clear the interrupts of active channels. This register can only clear edge triggered interrupts and will clear the corresponding bits in the status registers (IRQSR/FIQSR) and the polling register (IPR). Level triggered interrupts are not affected by this register and need to be cleared at the interrupt device. For example to clear Channel 3 interrupt (INT3) which has been programmed as edge triggered, the interrupt handler needs to write the value 0x0008 to ICLR. On the other hand to clear Channel 10 interrupt which is always level triggered, the interrupt handler needs to clear the interrupt in the UART unit. *For the watchdog timer interrupt, clearing bit 9 (CH12) will clear the interrupt but the system designer needs also to reset the watchdog timer counter to prevent a system reset as explained in the Watchdog Timer chapter.*

Prior to enabling edge triggered interrupts via IRQER or FIQER, it is recommended to write a '1' in the corresponding bit of the ICLR register to clear any prior status.

IRQ interrupt Enable Register (IRQER)

7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

15	14	13	12	11	10	9	8
///	///	///	///	CH11	CH10	CH9	CH8

Table 14-5.
IRQER Fields

FIELD NAME	DESCRIPTION
CHi	0: Disable Channel i IRQ Interrupt 1: Enable Channel i IRQ Interrupt
///	Reserved

The IRQ Interrupt Enable Register is used to enable/disable the IRQ interrupt for each interrupt channel. Disabling the IRQ interrupt for a channel will result in a value of '0' in the IRQ status register for that channel (IRQER(i) = 0 will produce IRQSR(i) = 0).

An interrupt channel can have either IRQER bit enabled or FIQER bit enabled. A channel can't have both interrupts enabled. Channel 12, Watchdog Timer Interrupt, is always enabled as an FIQ interrupt.

FIQ Interrupt Enable Register (FIQER)

7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

15	14	13	12	11	10	9	8
///	///	///	///	CH11	CH10	CH9	CH8

Table 14-6.
FIQER Fields

FIELD NAME	DESCRIPTION
CHi	0: Disable Channel i FIQ Interrupt 1: Enable Channel i FIQ Interrupt
///	Reserved

The FIQ Interrupt Enable Register is used to enable/disable the FIQ interrupt for each interrupt channel. Disabling the FIQ interrupt for a channel will result in a value of '0' in the FIQ status register for that channel (FIQER(i) = 0 will produce FIQSR(i) = 0).

An interrupt channel can have either IRQER bit enabled or FIQER bit enabled. A channel can't have both interrupts enabled. Channel 12, Watchdog Timer Interrupt, is always enabled as an FIQ interrupt.

IRQ Status Register (IRQSR)

7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

15	14	13	12	11	10	9	8
///	///	///	///	CH11	CH10	CH9	CH8

Table 14-7.
IRQSR Fields

FIELD NAME	DESCRIPTION
CHi	0: Channel i Interrupt is IRQ Disabled/Not Active 1: Channel i Interrupt is IRQ Enabled/Active
///	Reserved

The IRQ status register reflects the status of all channels enabled to produce an IRQ interrupt (IRQER(i) = 1). The channel interrupts are first ANDed with IRQER and then stored in this register. When an interrupt that is enabled for an IRQ occurs, the bit corresponding to the interrupt is set in IRQSR. The IRQ interrupt handler will examine this register to determine the channel(s) that caused the IRQ interrupt. If the IRQER bit for a channel is ‘0’, then the status bit for this channel will have a ‘0’. Note that Channel 12 is always a FIQ interrupt.

FIQ Status Register (FIQSR)

7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

15	14	13	12	11	10	9	8
///	///	///	CH12	CH11	CH10	CH9	CH8

Table 14-8.
FIQSR Fields

FIELD NAME	DESCRIPTION
CHi	0: Channel i Interrupt is FIQ Disabled/Not Active 1: Channel i Interrupt is FIQ Enabled/Active
///	Reserved

The FIQ Status Register reflects the status of all channels enabled to produce an FIQ interrupt (FIQER(i) = 1). The channel interrupts are first ANDed with FIQER and then stored in this register. When an interrupt that is enabled for an FIQ occurs, the bit corresponding to the interrupt is set in FIQSR. The FIQ interrupt handler will examine this register to determine the channel(s) that caused the FIQ interrupt. If the FIQER bit for a channel is '0', then the status bit for this channel will have a '0'. Note that CH12 is always a FIQ interrupt.

Interrupt Polling Register (IPR)

7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

15	14	13	12	11	10	9	8
///	///	///	CH12	CH11	CH10	CH9	CH8

Table 14-9.
IPR Fields

FIELD NAME	DESCRIPTION
CHi	0: Channel i Interrupt is Not Pending 1: Channel i Interrupt is Pending (IRQ or FIQ)
///	Reserved

IPR reflects the status of all interrupt channels. For edge triggered interrupts, the latched value is reflected in this register. For level triggered interrupts, the actual value is reflected in this register. The input to this register is taken prior to applying IRQER/FIQER. A pending interrupt could be either an IRQ or FIQ depending on the configuration of the interrupt channel.

PRIORITY

The ARM7DI has two levels of interrupt priorities:

- High Priority Interrupt (FIQ) with an interrupt vector at 0x0000001C
- Low Priority Interrupt (IRQ) with an interrupt vector at 0x00000018

Any interrupt can be enabled as either FIQ or IRQ via FIQER/IRQER registers. When multiple FIQ(IRQ) interrupts are active at the same time, the OR of all FIQ (IRQ) is sent to the ARM7DI. The interrupt handler needs to assign its own priority. The interrupt handler has access to the status registers (FIQSR, IRQSR) to determine the source(s) of the interrupt. The interrupt handler can then determine which interrupt to service first.

EXCEPTIONS

The ARM7DI core has other exceptions besides FIQ and IRQ. Table 14-10 summarizes these exceptions as defined in the ARM7DI data sheet. For more detailed information on these exceptions, refer to the ARM7DI data sheet.

Table 14-10.
Exception Vectors Summary

VECTOR ADDRESS	EXCEPTION	ARM7DI OPERATING MODE	PRIORITY
0x00000000	Reset	Supervisor	1 (highest)
0x00000004	Undefined Instruction[1]	Undefined	6 (lowest)
0x00000008	Software Interrupt[1]	Supervisor	6 (lowest)
0x0000000C	Abort (prefetch)	Abort	5
0x00000010	Abort (data)[2]	Abort	2
0x00000014	Reserved		
0x00000018	IRQ	IRQ	4
0x0000001C	FIQ	FIQ	3

NOTES:

1. Mutually Exclusive.
2. If a data abort occurs at the same time as a FIQ, and FIQ is enabled, ARM7DI core will enter the data abort handler and then immediately proceed to the FIQ vector. A normal return from FIQ will cause the data abort handler to resume execution.

Chapter 15

I/O CONFIGURATION

INTRODUCTION

The 790A has a 16-bit Input/Output Configuration Register (IOCR) that allows the system designer to program multi-functional pins and control the GATE signals to the Counter/Timers.

REGISTER MAP

The base address for I/O Configuration register(s) is 0xFFFFA410. The offset, initial value, access and number of bits for each register are as follows.

Table 15-1.
I/O Configuration Register Map

IOCR REGISTER	ADDRESS OFFSET [7:0]	RESET VALUE	ACCESS	NUMBER OF BITS
IOCR	0x00	0x000000	R/W	16

REGISTER DESCRIPTION

Input/Output Configuration Register (IOCR)

7	6	5	4	3	2	1	0
PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
15	14	13	12	11	10	9	8
////	CT2G		CT1G		CT0G		PU8

Table 15-2.
IOCR Fields

BIT POSITION	NAME	DESCRIPTION
0	PU0*	Configure PB2/$\overline{\text{RI1}}$ pin: 0 ---> PB2 (input/output) 1 ---> $\overline{\text{RI1}}$ (input)
1	PU1*	Configure PB3/$\overline{\text{CTS1}}$ pin: 0 ---> PB3 (input/output) 1 ---> $\overline{\text{CTS1}}$ (input)
2	PU2*	Configure PB4/$\overline{\text{CTS0}}$ pin: 0 ---> PB4 (input/output) 1 ---> $\overline{\text{CTS0}}$ (input)

Table 15-3.
IOCR Fields

BIT POSITION	NAME	DESCRIPTION
3	PU3*	Configure PB5/$\overline{RI0}$ pin: 0 ---> PB5 (input/output) 1 ---> $\overline{RI0}$ (input)
4	PU4*	Configure PB6/$\overline{DCD0}$ pin: 0 ---> PB6 (input/output) 1 ---> $\overline{DCD0}$ (input)
5	PU5*	Configure PB7/$\overline{DSR0}$ pin: 0 ---> PB7 (input/output) 1 ---> $\overline{DSR0}$ (input)
6	PU6*	Configure PC0/$\overline{RTS1}$ pin: 0 ---> PC0 (input/output) 1 ---> $\overline{RTS1}$ (output)
7	PU7*	Configure PC1/ $\overline{RTS0}$ pin: 0 ---> PC1 (input/output) 1 ---> $\overline{RTS0}$ (output)
8	PU8*	Configure PC2/$\overline{DTR0}$ pin: 0 ---> PC2 (input/output) 1 ---> $\overline{DTR0}$ (output)
10:9	CT0G	Counter/Timer0 Gate Source: 00 ---> External Gate pin CTGATE0 01 ---> PWM0 Output (same value on PWM0 pin) 10 ---> Logic '0' 11 ---> Logic '1'
12:11	CT1G	Counter/Timer1 Gate Source: 00 ---> External Gate pin CTGATE1 01 ---> PWM1 Output (same value on PWM1 pin) 10 ---> Logic '0' 11 ---> Logic '1'
14:13	CT2G	Counter/Timer2 Gate Source: 00 ---> External Gate pin CTGATE2 01 ---> PWM2 Output (same value on PWM2 pin) 10 ---> Logic '0' 11 ---> Logic '1'
15	////	unused

NOTE:*

1. PB[7:2] and PC[2:0] can be programmed on a bit by bit basis to either function as a PPI signal or a modem signal
2. If *any* of PB[7:2] signals are programmed as modem signals, all of PB[7:0] must be programmed as input via PPI Control Register PPI_CTLR.
3. If any of PC[1:0] signals are programmed as modem signals, all of PC [1:0] must be programmed as output via PPI control register PPI-CTLR
4. If PC[2] signal is programmed as a modem signal, PC[2] must be programmed as output via PPI Control Register PPI_CTLR.

Chapter 16

RESET

INTRODUCTION

The 790A can be reset from an external reset source, or from the on-chip Watchdog Timer. It can also reset external devices from the on-chip Watchdog Timer or via software.

EXTERNAL RESET SOURCE

The 790A and the JTAG TAP Controller are externally reset by asserting $\overline{\text{RESETI}}$ LOW. $\overline{\text{RESETI}}$ is an asynchronous input and is sampled on the rising edge of the system clock, XCLK. Once sampled LOW, $\overline{\text{RESETI}}$ must remain active LOW for 8.5 cycles. Upon system power-up, VCC should be stable before $\overline{\text{RESETI}}$ is recognized by the 790A. The 790A internal reset logic has a built-in glitch detector that verifies that $\overline{\text{RESETI}}$ is driven LOW for three consecutive XCLK cycles before resetting the chip (valid reset). $\overline{\text{RESETI}}$ is a level sensitive signal. A LOW level will cause the instruction being executed to terminate abnormally and the ARM7DI core will perform dummy instruction fetches with the address incrementing from the point where $\overline{\text{RESETI}}$ was activated. When $\overline{\text{RESETI}}$ becomes HIGH for at least one clock, the ARM7DI will restart from address 0x00000000 in supervisor mode. The 790A will sample and capture $\overline{\text{BB}}$ on the rising edge of $\overline{\text{RESETI}}$ to determine if the boot memory is either x8 or x16. It is recommended that $\overline{\text{BB}}$ be permanently tied LOW for x8 boot memory or HIGH for x16 boot memory. The captured state of $\overline{\text{BB}}$ will be used when the Watchdog Timer generates a system reset.

The 790A has a reset output pin, $\overline{\text{RESETO}}$, which is driven LOW as long as $\overline{\text{RESETI}}$ is driven LOW after validating $\overline{\text{RESETI}}$. So $\overline{\text{RESETO}}$ is driven LOW 3.5 cycles after $\overline{\text{RESETI}}$ LOW is sampled.

The JTAG TAP Controller reset signal ($\overline{\text{TRST}}$) is internally connected to $\overline{\text{RESETI}}$. The IEEE 1149.1 - 1990 standard requires that JTAG inputs be pulled up to a good logic level to achieve normal operations.

WATCHDOG TIMER

The Watchdog Timer can be programmed to cause two types of reset when it time-outs.

External Reset: The 790A will drive $\overline{\text{RESETO}}$ LOW for 8 cycles to cause external reset only. The 790A is not affected.

System Reset: The 790A will reset internally and drive $\overline{\text{RESETO}}$ LOW for 8 cycles to cause external reset. The 790A uses the captured state of $\overline{\text{BB}}$ to boot up.

SOFTWARE CONTROLLED RESET

The 790A provides a mechanism for software to activate/deactivate $\overline{\text{RESETO}}$. A write only 1-bit register, SWRST, residing at address 0xFFFFAC38, will control $\overline{\text{RESETO}}$ as follows:

Table 16-1.
SWRST Fields

SWRST	DESCRIPTION
0	$\overline{\text{RESETO}}$ is driven HIGH
1	$\overline{\text{RESETO}}$ is driven LOW. It will remain LOW until a logical '0' is written into SWRST.

NOTE:
1. The Reset Value of SWRST is '0'.
2. When software sets SWRST register to '1' to assert $\overline{\text{RESETO}}$, DRAM refresh will stop. DRAM refresh can't be activated while SWRST register is set to '1'. When SWRST is set to '0', DRAM Refresh Register (DRR) must be written again with the refresh value to activate refresh.

JTAG RESET

$\overline{\text{TRST}}$ needs to be held LOW, then HIGH to ensure that the JTAG TAP controller is reset. The IEEE 1149.1 - 1990 standard requires that JTAG inputs be pulled up to good logic levels to achieve normal operations.

Since $\overline{\text{TRST}}$ is an optional signal according to the IEEE Standard 1149.1 - 1990, the 790A internally connects $\overline{\text{TRST}}$ and $\overline{\text{RESETI}}$. This will ensure proper reset for both the 790A logic and 790A JTAG TAP controller when the 790A is externally reset. A clock on JTAG clock, TCK, is not necessary to reset the JTAG TAP controller. No external pin is dedicated for $\overline{\text{TRST}}$.

Chapter 17

IDENTIFICATION REGISTER

INTRODUCTION

The identification register, IDR, indicates the family name, member name, and production revision associated with the 790A. Software can take advantage of this register to identify what processor it is running on and its production revision.

NOTE: This register is different than the ID Code Register in the ARM7DI core that is accessible through the JTAG interface only.

REGISTER MAP

The base address for Identification Register is 0xFFFFA40C. The offset, initial value, access and number of bits for each register is as follows:

Table 17-1
Register Map

ID REGISTER	ADDRESS OFFSET [7:0]	RESET VALUE	ACCESS	NUMBER OF BITS
IDR	0x00	0x2100*	Read	16

NOTE: *For first production parts. Bits [5:0] depends on the production level. Bits [15:6] are constant.

REGISTER DESCRIPTION

Identification Register (IDR)

15 13	12	11 6	5 0
FAM	T	MEM	PROD

The IDR is a 16-bit read only register. The register consists of the following fields:

Table 17-2
IDR Fields

BIT POSITION	NAME	DESCRIPTION
5:0	PROD	Production Revision: 00000: Revision 0, First Production 00001: Revision 1, Second Production :
11:6	MEM	Member Name 000100: LH77790A
12	T	Thumb Aware 0: Does Not Support Thumb Code 1: Supports Thumb Code
15 - 13	FAM	Family Name: 001: Summit Family (LH77790A)

The LH77790A is a member of the Summit family and it does not support Thumb code.

Chapter 18

DEBUG INTERFACE

INTRODUCTION

The debug interface for the 790A is a JTAG-style serial interface. It is based on the IEEE standard 1149.1 - 1990, 'Standard Test Access Port and Boundary-Scan Architecture'. The scan cells are not fully JTAG compliant (e.g. no update stage). The *ARM7DI Data Sheet* describe the limitations on their use.

BLOCK DIAGRAM

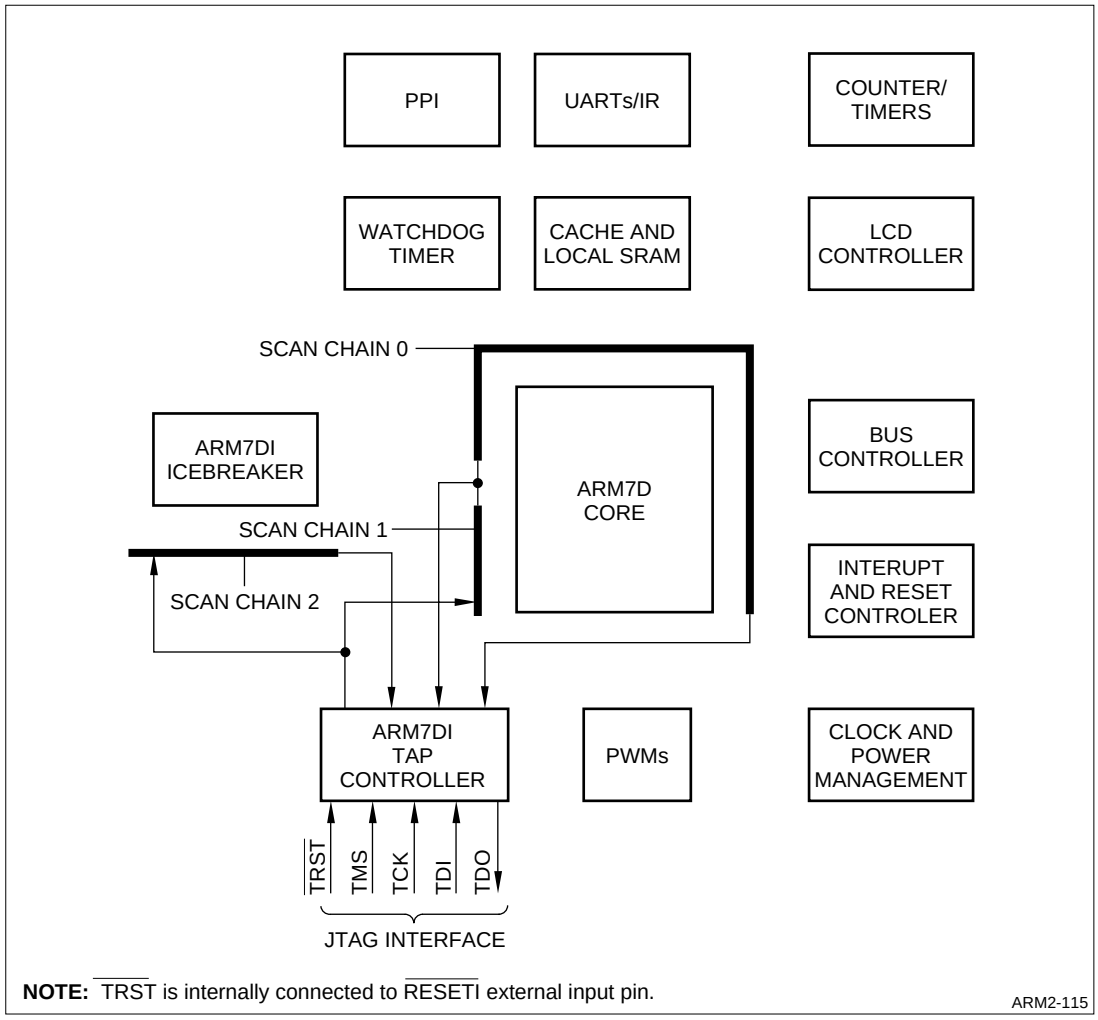


Figure 18-1. LH77790A Scan Chains

GENERAL OPERATION

The 790A JTAG interface consists of four signals: TCK, TMS, TDI, and TDO. $\overline{\text{TRST}}$ is connected internally to the $\overline{\text{RESETI}}$ input to allow resetting the tap controller when the 790A is externally reset. The JTAG interface allows access to ARM7DI scan chains (scan chains 0, 1, 2) as shown in figure 18-1. The 790A has no boundary scan chain. The JTAG interface also allows the 790A target system to directly connect to ARM's EmbeddedICE as shown in figure 18-2.

Scan chains 0, 1, 2 are detailed in the *ARM7DI Data Sheet.*

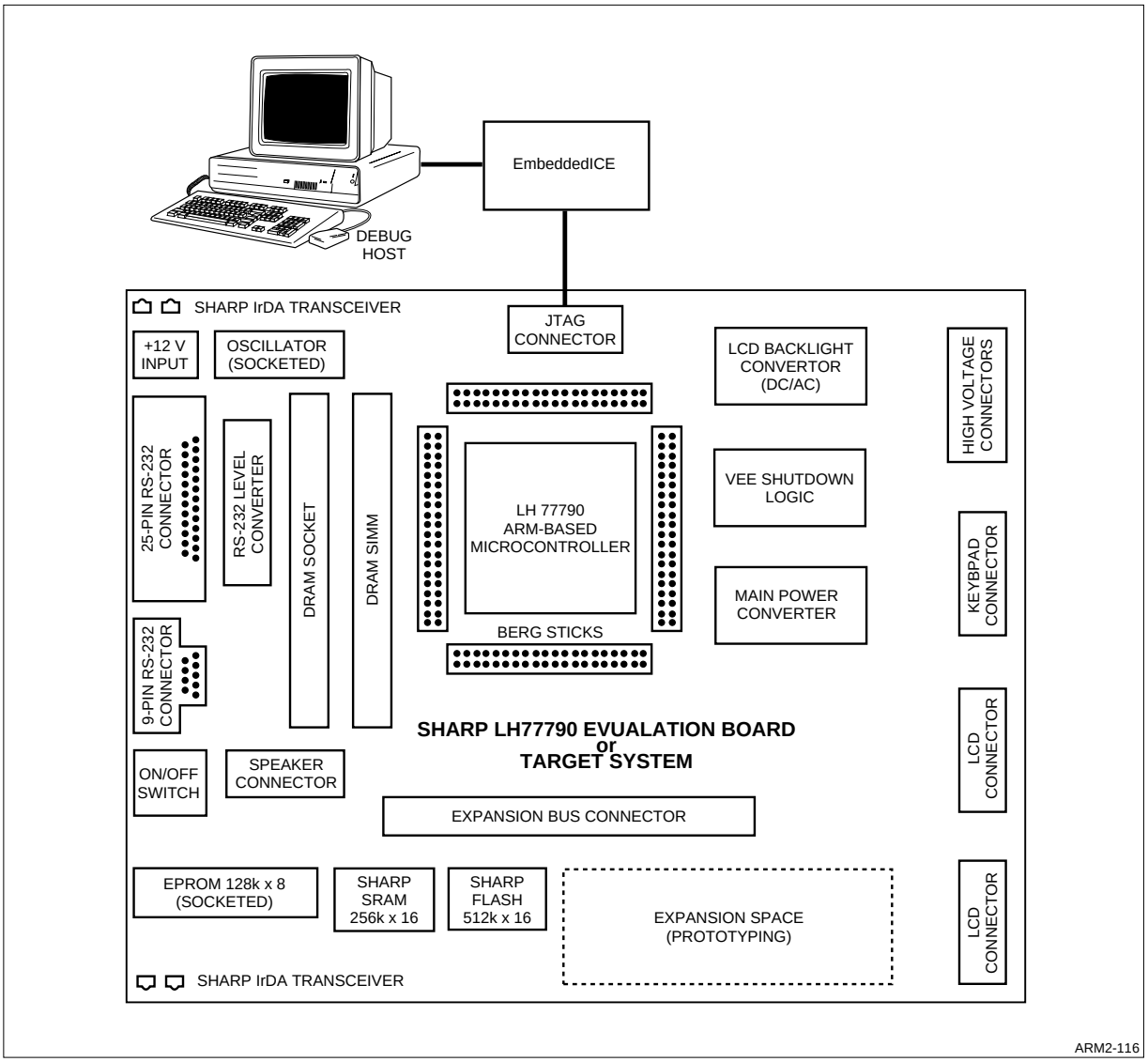


Figure 18-2. LH77790A Target System Connection to ARM's Embedded ICE via JTAG

Pullup Resistors

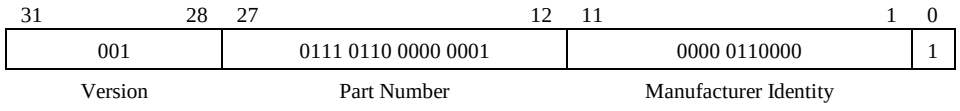
The IEEE 1149.1- 1990 standard requires that the JTAG inputs be pulled up to good logic levels to achieve normal operations. Table 18-3 shows a recommended pull-up resistors' values that is also compatible with ARM's EmbeddedICE. $\overline{TRST}$ is internally connected to $\overline{RESETI}$ input pin.

Table 18-3.
Recommended JTAG Pull-up Resistors

JTAG INPUT SIGNAL	PULL-UP RESISTOR
TDI	56 kΩ
TMS	56 kΩ
TCK	56 kΩ

Device Identification Code

The ARM7DI provides a 32-bit device identification code that is accessible only through the JTAG interface. The format of the device identification code associated with the LH77790A is 0x17601061.



JTAG Reset

A scan chain must be selected prior to usage.

Chapter 19

MEMORY MAP & REGISTER SUMMARY

MEMORY MAP

Table 19-1 summarizes the memory map of the LH77790A (790).

Table 19-1.
The LH77790A Memory Map

MEMORY ADDRESS (HEX)	DESCRIPTION
0x00000000	Reset Vector
0x00000004	Undefined Instruction Vector
0x00000008	Software Interrupt (SWI) Vector
0x0000000C	Abort (Prefetch)Vector
0x00000010	Abort (Data) Vector
0x00000014	Reserved
0x00000018	IRQ Interrupt Vector
0x0000001C	FIQ Interrupt Vector
0x00000000 – 0x000007FF or 0x60000000 – 0x600007FF	Local SRAM, LSCR[1] = 0 Local SRAM, LSCR[1] = 1
0x60000800 – 0x60000FFF	Cache in SRAM Mode [Data RAM]
0x60001000 – 0x600017FF	Reserved
0xFFFF0000 – 0xFFFF9FFF	Internal Peripherals Region
0xFFFFA000 – 0xFFFFFFFF	System Configuration Region

Table 19-2 details the address range and base address for each group of registers. The offset for each register within a group is outlined in each group's section and is also shown in the Register Summary (Table 19-2).

Table 19-2.
Address Range and Base Address of Registers

GROUP	RANGE	BASE
System Configuration Region		
Memory Segment Registers	0xFFFFA000 – 0xFFFFA0FF	0xFFFFA000
Bank Configuration Registers	0xFFFFA100 – 0xFFFFA1FF	0xFFFFA100
Reserved	0xFFFFA200 – 0xFFFFA3FF	0xFFFFA200
Cache Control Register	0xFFFFA400 – 0xFFFFA400	0xFFFFA400
Local SRAM Control Register	0xFFFFA404 – 0xFFFFA404	0xFFFFA404
Reserved	0xFFFFA408 – 0xFFFFA408	0xFFFFA408
Identification Register	0xFFFFA40C – 0xFFFFA40C	0xFFFFA40C
I/O Configuration Registers	0xFFFFA410 – 0xFFFFA410	0xFFFFA410
LCD Bit Control Register	0xFFFFA414 – 0xFFFFA414	0xFFFFA414
Interrupt Controller Registers	0xFFFFA800 – 0xFFFFABFF	0xFFFFA800
Clock and Power Management/ Watchdog Timer/SW Reset Registers	0xFFFFAC00 – 0xFFFFAFFF	0xFFFFAC00
Reserved	0xFFFFB000 – 0xFFFFFFFF	0xFFFFB000
Internal Peripherals Region		
UART0 Registers	0xFFFF0000 – 0xFFFF03FF	0xFFFF0000
UART1 Registers	0xFFFF0400 – 0xFFFF07FF	0xFFFF0400
UART2 Registers	0xFFFF0800 – 0xFFFF0BFF	0xFFFF0800
SIR Registers	0xFFFF0C00 – 0xFFFF0FFF	0xFFFF0C00
PWM Registers	0xFFFF1000 – 0xFFFF13FF	0xFFFF1000
LCD Controller Registers	0xFFFF1400 – 0xFFFF17FF	0xFFFF1400
Counter/Timer Registers	0xFFFF1800 – 0xFFFF1BFF	0xFFFF1800
PPI Registers	0xFFFF1C00 – 0xFFFF1FFF	0xFFFF1C00
Reserved	0xFFFF2000 – 0xFFFF9FFF	0xFFFF2000

System Configuration Region

This 24KB region is dedicated for System Configuration Registers and should not be mapped to external devices. Even though the region is partially assigned, future embedded microcontrollers will use unassigned memory for future system configuration registers. This region can be included as part of any of the eight segments (segments 0-7) or default segment. Only the privilege bits (SPR, UPR) from the SDR for that segment are recognized when an address belongs to this region of 24KB. Other bits like BSEL, C and HW are ignored for addresses in this region only. This gives the system designer the

flexibility to assign a privilege to this region. This region is reserved for the following System Configuration registers:

- Segment Descriptor Registers (SDRs)
- START/STOP Registers
- Bank Configuration Registers (BCRs)
- DRAM Refresh Register (DRR)
- Cache Control Registers (CCRs)
- Local SRAM Control Registers (LSCRs)
- Identification Register (IDR)
- I/O Configuration Registers (IOCRs)
- LCD Bit Control Register (LCD_BITCTL)
- Interrupt Controller Registers
- Clock and Power Management Registers
- Watchdog Timer Registers

Internal Peripherals Region

This 40KB region is dedicated for internal peripheral registers and should not be mapped to external devices. Even though the region is partially assigned, future embedded microcontrollers will use unassigned memory for future peripheral registers. This region can be included as part of any of the eight segments (segments 0 - 7) or it will use the default segment. Only the privilege bits (SPR, UPR) from the SDR for that segment are recognized when an address belongs to this region (40KB). Other bits like BSEL, C and HW are ignored for addresses in this region only. This gives the system designer the flexibility to assign a privilege to this region. The base address for each group will be on a 1KB boundary.

Cache Region

When the Cache is operating in SRAM mode, the Data RAM (512 words) will map to address locations 60000800-60000FFF. In the SRAM mode, the cache can be included as part of any of the eight segments (segments 0 - 7) or it will use the default segment. Only the privilege bits (SPR, UPR) from the SDR for that segment are recognized when an address belongs to this region.

Local SRAM Region

The local SRAM can be programmed to map to two different regions as shown in the above table. The mapping is controlled by the Local SRAM Control Register (LSCR) as discussed in the local SRAM section. Mapping the local SRAM to the lower region allows the exception vectors to be mapped to the local SRAM for faster access. This region can be included as part of any of the eight segments (segments 0 - 7) or it will use the default segment. Only the privilege bits (SPR, UPR) from the SDR for that segment are recognized when an address belongs to this region.

REGISTER SUMMARY*

Table 19-3.
LH77790A Register Summary

REGISTER	NAME	ADDRESS	RESET VALUE	ACCESS	SIZE
Cache					
CCR	Cache Control	0xFFFFA400	0x00	R/W	8
Local SRAM					
LSCR	Local SRAM Control	0xFFFFA404	0x02	R/W	8
Memory & Peripherals					
START0	Segment 0 START	0xFFFFA000	0x00000000	R/W	32
START1	Segment 1 START	0xFFFFA004	0x00000000	R/W	32
START2	Segment 2 START	0xFFFFA008	0x00000000	R/W	32
START3	Segment 3 START	0xFFFFA00C	0x00000000	R/W	32
START4	Segment 4 START	0xFFFFA010	0x00000000	R/W	32
START5	Segment 5 START	0xFFFFA014	0x00000000	R/W	32
START6	Segment 6 START	0xFFFFA018	0x00000000	R/W	32
START7	Segment 7 START	0xFFFFA01C	0x00000000	R/W	32
STOP0	Segment 0 STOP	0xFFFFA020	0x00000000	R/W	32
STOP1	Segment 1 STOP	0xFFFFA024	0x00000000	R/W	32
STOP2	Segment 2 STOP	0xFFFFA028	0x00000000	R/W	32
STOP3	Segment 3 STOP	0xFFFFA02C	0x00000000	R/W	32
STOP4	Segment 4 STOP	0xFFFFA030	0x00000000	R/W	32
STOP5	Segment 5 STOP	0xFFFFA034	0x00000000	R/W	32
STOP6	Segment 6 STOP	0xFFFFA038	0x00000000	R/W	32
STOP7	Segment 7 STOP	0xFFFFA03C	0x00000000	R/W	32
SDR0	Segment 0 Descriptor	0xFFFFA040	0x0000	R/W	16
SDR1	Segment 1 Descriptor	0xFFFFA044	0x0000	R/W	16
SDR2	Segment 2 Descriptor	0xFFFFA048	0x0000	R/W	16
SDR3	Segment 3 Descriptor	0xFFFFA04C	0x0000	R/W	16
SDR4	Segment 4 Descriptor	0xFFFFA050	0x0000	R/W	16
SDR5	Segment 5 Discriptor	0xFFFFA054	0x0000	R/W	16
SDR6	Segment 6 Descriptor	0xFFFFA058	0x0000	R/W	16
SDR7	Segment 7 Descriptor	0xFFFFA05C	0x0000	R/W	16
SDR8	Segment 8 Descriptor	0xFFFFA060	0x0000	R/W	16

NOTE: *All registers that are more than 8-bit wide SHOULD be treated as a 32-bit register. Otherwise, the compiler will access these registers using two STRB/LDRB instructions (Store/Load Bytes) resulting in incorrent data being written or read.

Table 19-3.
LH77790A Register Summary (cont'd)

REGISTER	NAME	ADDRESS	RESET VALUE	ACCESS	SIZE
BCR0	Bank Control 0 (SRAM)	0xFFFFA100	8-bit Boot: x7003 ($\overline{BB}$ = 0) 16-bit Boot: xF009 ($\overline{BB}$ = 1)	R/W	16
BCR1	Bank Control 1 (SRAM)	0xFFFFA104	0x0000	R/W	16
BCR2	Bank Control 2 (SRAM)	0xFFFFA108	0x0000	R/W	16
BCR3	Bank Control 3 (SRAM)	0xFFFFA10C	0x0000	R/W	16
BCR4	Bank Control 4 (SRAM)	0xFFFFA110	0x0000	R/W	16
BCR5	Bank Control 5 (SRAM)	0xFFFFA114	0x0000	R/W	16
BCR6a	Bank Control 6a (DRAM)	0xFFFFA118	0x0000	R/W	16
BCR7a	Bank Control 7a (DRAM)	0xFFFFA11C	0x0000	R/W	16
BCR6b	Bank Control 6b (DRAM)	0xFFFFA120	0x0000	R/W	16
BCR7b	Bank Control 7b (DRAM)	0xFFFFA124	0x0000	R/W	8
DRR	DRAM Refresh	0xFFFFA128	0x0000	R/W	16
UART0					
RBR0 THR0 DLL0	Receiver Buffer Transmitter Holding Divisor Latch LSB	0xFFFF0000 0xFFFF0000 0xFFFF0000	Undefined Undefined Undefined	R W R/W	8 8 8
IER0 DLM0	Interrupt Enable Divisor Latch MSB	0xFFFF0004 0xFFFF0004	0x00 Undefined	R/W R/W	8 8
IIR0	Interrupt Identification	0xFFFF0008	0x01	R	8
LCR0	Line Control	0xFFFF000C	0x00	R/W	8
MCR0	Modem Control	0xFFFF0010	0x00	R/W	8
LSR0	Line Status	0xFFFF0014	0x60	R	8
MSR0	Modem Status	0xFFFF0018	MSR[3:0]=0 MSR[7:4] are inputs	R/W	8
SCR0	Scratch Pad	0xFFFF001C	Undefined	R/W	8
UART1					
RBR1 THR1 DLL1	Receiver Buffer Transmitter Holding Divisor Latch LSB	0xFFFF0400 0xFFFF0400 0xFFFF0400	Undefined Undefined Undefined	R W R/W	8 8 8
IER1 DLM1	Interrupt Enable Divisor Latch MSB	0xFFFF0404 0xFFFF0404	0x00 Undefined	R/W R/W	8 8
IIR1	Interrupt Identification	0xFFFF0408	0x01	R	8
LCR1	Line Control	0xFFFF040C	0x00	R/W	8
MCR1	Modem Control	0xFFFF0410	0x00	R/W	8

Table 19-3.
LH77790A Register Summary (cont'd)

REGISTER	NAME	ADDRESS	RESET VALUE	ACCESS	SIZE
LSR1	Line Status	0xFFFF0414	0x60	R	8
MSR1	Modem Status	0xFFFF0418	MSR[3:0] = 0 MSR[7:4] are inputs	R/W	8
SCR1	Scratch Pad	0xFFFF041C	Undefined	R/W	8
UART2					
RBR2 THR2 DLL2	Receiver Buffer Transmitter Holding Divisor Latch LSB	0xFFFF0800 0xFFFF0800 0xFFFF0800	Undefined Undefined Undefined	R W R/W	8 8 8
IER2 DLM2	Interrupt Enable Divisor Latch MSB	0xFFFF0804 0xFFFF0804	0x00 Undefined	R/W R/W	8 8
IIR2	Interrupt Identification	0xFFFF0808	0x01	R	8
LCR2	Line Control	0xFFFF080C	0x00	R/W	8
MCR2	Modem Control	0xFFFF0810	0x00	R/W	8
LSR2	Line Status	0xFFFF0814	0x60	R	8
MSR2	Modem Status	0xFFF0818	MSR[3:0] = 0 MSR[7:4] are Input	R/W	8
SCR2	Scratch Pad	0xFFFF081C	Undefined	R/W	8
SIR (IrDA/DASK)					
SIRCTLR	SIR Control	0xFFFF0C00	0x02	R/W	8
PWM					
PWM0_TC	PWM0 Terminal Count	0xFFFF1000	0x00	W	8
PWM0_DC	PWM0 Duty Cycle	0xFFFF1004	0x00	W	8
PWM0_ENB	PWM0 Enable	0xFFFF1008	0x00	W	8
PWM0_DIV	PWM0 Divide Value	0xFFFF100C	0x00	W	8
PWM0_SYNC	PWM0 Synchronous	0xFFFF1010	0x00	W	8
PWM0_INV	PWM0 Invert	0xFFFF1014	0x00	W	8
PWM0_UPDT	PWM0 Update	0xFFFF1018	0x00	W	8
PWM1_TC	PWM1 Terminal Count	0xFFFF1020	0x00	W	8
PWM1_DC	PWM1 Duty Cycle	0xFFFF1024	0x00	W	8
PWM1_ENB	PWM1 Enable	0xFFFF1028	0x00	W	8
PWM1_DIV	PWM1 Divide Value	0xFFFF102C	0x00	W	8
PWM1_SYNC	PWM1 Synchronous	0xFFFF1030	0x00	W	8
PWM1_INV	PWM1 Invert	0xFFFF1034	0x00	W	8
PWM1_UPDT	PWM1 Update	0xFFFF1038	0x00	W	8
PWM2_TC	PWM2 Terminal Count	0xFFFF1040	0x0000	W	16
PWM2_DC	PWM2 Duty Cycle	0xFFFF1044	0x0000	W	16

Table 19-3.
LH77790A Register Summary (cont'd)

REGISTER	NAME	ADDRESS	RESET VALUE	ACCESS	SIZE
PWM2_ENB	PWM2 Enable	0xFFFF1048	0x00	W	8
PWM2_DIV	PWM2 Divide Value	0xFFFF104C	0x00	W	8
PWM2_SYNC	PWM2 Synchronous	0xFFFF1050	0x00	W	8
PWM2_INV	PWM2 Invert	0xFFFF1054	0x00	W	8
PWM2_UPDT	PWM2 Update	0xFFFF1058	0x00	W	8
PWMA_TC	All TWMs Terminal Count	0xFFFF1060	0x0000	W	16
PWMA_DC	All PWMs Duty Cycle	0xFFFF1064	0x0000	W	16
PWMA_ENB	All PWMs Enable	0xFFFF1068	0x00	W	8
PWMA_DIV	All PWMs Divide Value	0xFFFF106C	0x00	W	8
PWMA_SYNC	All PWMs Synchronous	0xFFFF1070	0x00	W	8
PWMA_INV	All PWMs Invert	0xFFFF1074	0x00	W	8
PWMA_UPDT	All PWMs Update	0xFFFF1078	0x00	W	8
LCD CONTROLLER					
LCD_MODE	Operation Mode	0xFFFF1400	0x00	W	8
LCD_BC	Line Display Byte Count	0xFFFF1404	0x00	W	8
LCD_CP1W	Line Pulse Width	0xFFFF1408	0x00	W	8
LCD_DUTY[7:0]	Duty Cycle	0xFFFF140C	0x00	W	8
LCD_DUTY[9:8]		0xFFFF1410	0x00	W	8
LCD_SADR1[7:0]	Screen #1 Frame Buffer	0xFFFF1414	0x00	W	8
LCD_SADR1[15:8]	Start Address	0xFFFF1418	0x00	W	8
LCD_SADR1[23:16]		0xFFFF141C	0x00	W	8
LCD_SADR1[31:24]		0xFFFF1420	0x00	W	8
LCD_SADR2[7:0]	Screen #2 Frame Buffer	0xFFFF1424	0x00	W	8
LCD_SADR2[15:8]	Start Address	0xFFFF1428	0x00	W	8
LCD_SADR2[23:16]		0xFFFF142C	0x00	W	8
LCD_SADR2[31:24]		0xFFFF1430	0x00	W	8
LCD_VLC1[7:0]	Screen #1 Vertical	0xFFFF1434	0x00	W	8
LCD_VLC1[9:8]	Line Count	0xFFFF1438	0x00	W	8
LCD_VDLT	Virtual Display Delta	0xFFFF143C	0x00	W	8
LCD_GRAY1	Gray Shade 1	0xFFFF1440	0x00	W	8
LCD_GRAY2	Gray Shade 2	0xFFFF1444	0x00	W	8
LCD_CLKDIV	Clock Divider	0xFFFF1448	0x00	W	8
LCD_MCLKW[7:0]	MCLK Width	0xFFFF144C	0x00	W	8
LCD_MCLKW[9:8}		0xFFFF1450	0x00	W	8
LCD_BITCTL	LCD Bit Control	0xFFFFA414	0x00	R/W	8

Table 19-3.
LH77790A Register Summary (cont'd)

REGISTER	NAME	ADDRESS	RESET VALUE	ACCESS	SIZE
Counter/Timers					
CT_CNTR0	Counting Element 0	0xFFFF1800	Undefined	R/W	8
CT_CNTR1	Counting Element 1	0xFFFF1804	Undefined	R/W	8
CT_CNTR2	Counting Element 2	0xFFFF1808	Undefined	R/W	8
CT_CWR	Counter/Timers Control	0xFFFF180C	Undefined	W	8
PPI					
PPI_PA	Port A	0xFFFF1C00	Input/Mode 0	R/W	8
PPI_PB	Port B	0xFFFF1C04	Input/Mode 0	R/W	8
PPI_PC	Port C	0xFFFF1C08	Input/Mode 0	R/W	8
PPI_CTLR	PPI Control	0xFFFF1C0C	0b0011011	W	8
Interrupt Controller					
ICR0	Configuration 0	0xFFFFA800	0x00	R/W	16
ICR1	Configuration 1	0xFFFFA804	0x00	R/W	8
ICLR	Clear	0xFFFFA808	0x00	W	16
IRQER	IRQ Enable	0xFFFFA80C	0x00	R/W	16
FIQER	FIQ Enable	0xFFFFA810	0x00	R/W	16
IRQSR	IRQ Status	0xFFFFA814	0x00	R	16
FIQSR	FIQ Status	0xFFFFA818	0x00	R	16
IPR	Polling	0xFFFFA81C	0x00	R	16
Clock & Power Management					
PCSR	Peripheral Clock Select	0xFFFFAC04	0x00	R/W	9
U0CCR	UART0 Clock Control	0xFFFFAC08	0x01	R/W	9
U1CCR	UART1 Clock Control	0xFFFFAC0C	0x01	R/W	9
U2CCR	UART2 Clock Control	0xFFFFAC10	0x01	R/W	9
CT0CCR	Counter/Timer0 Clock Control	0xFFFFAC18	0x01	R/W	9
CT1CCR	Counter/Timer1 Clock Control	0xFFFFAC1C	0x01	R/W	9
CT2CCR	Counter/Timer2 Clock Control	0xFFFFAC20	0x01	R/W	9
CCCR	CPU Clock Control	0xFFFFAC28	0x01	R/W	8
Watchdog Timer					
WDCTLR	WDT Control	0xFFFFAC30	0x00	R/W	8
WDCR	WDT Counter	0xFFFFAC34	0x00000000	W	32
I/O Configuration					
IOCR	IO Configuration	0xFFFFA410	0x0000	R/W	16
Reset					
SWRST	Software Reset	0xFFFFAC38	0	W	1

Chapter 20
PACKAGE SPECIFICATION

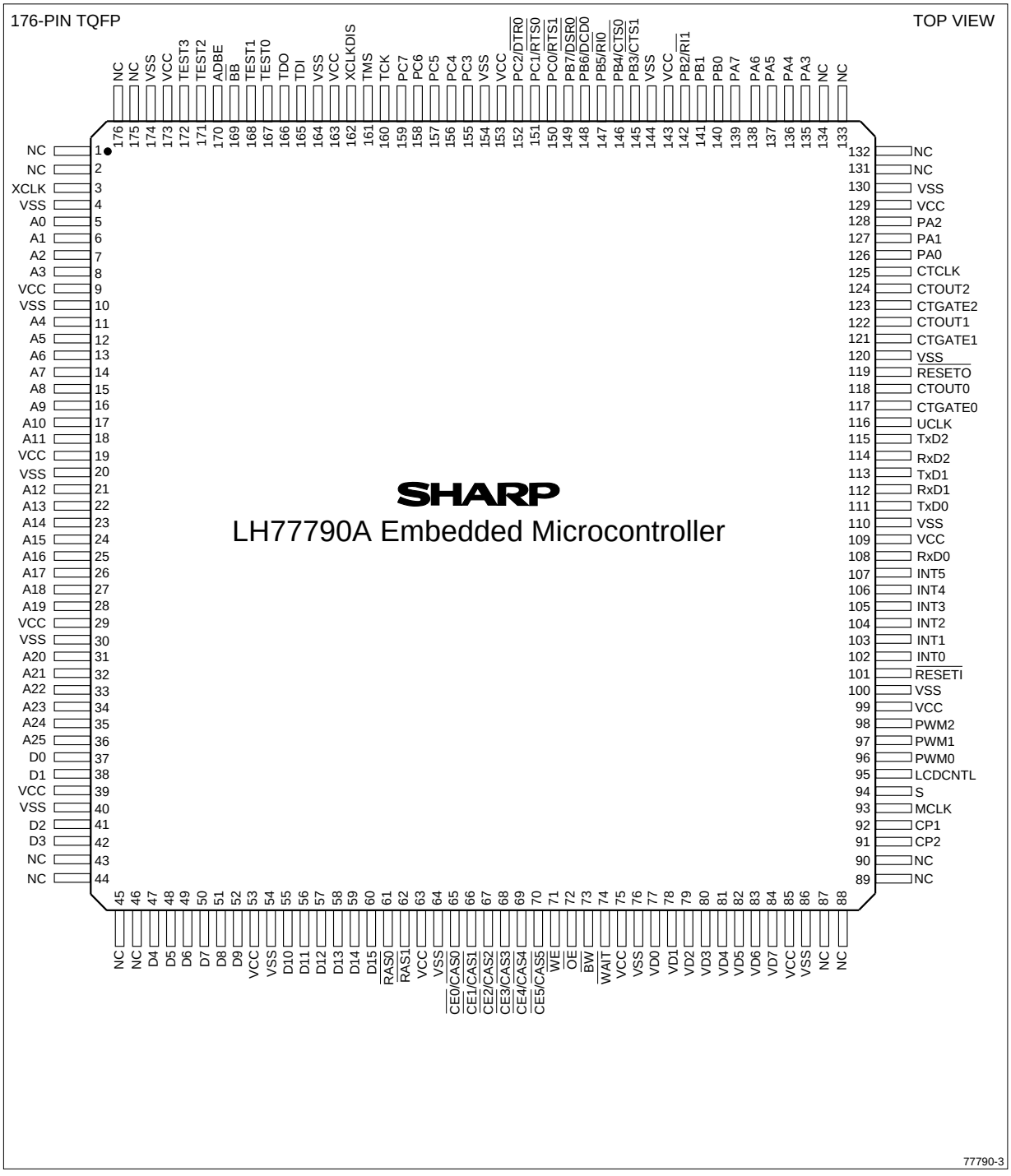


Figure 20-1. LH77790A 176-Lead TQFP (Thin Quad Flat Pack) Pin Assignment.

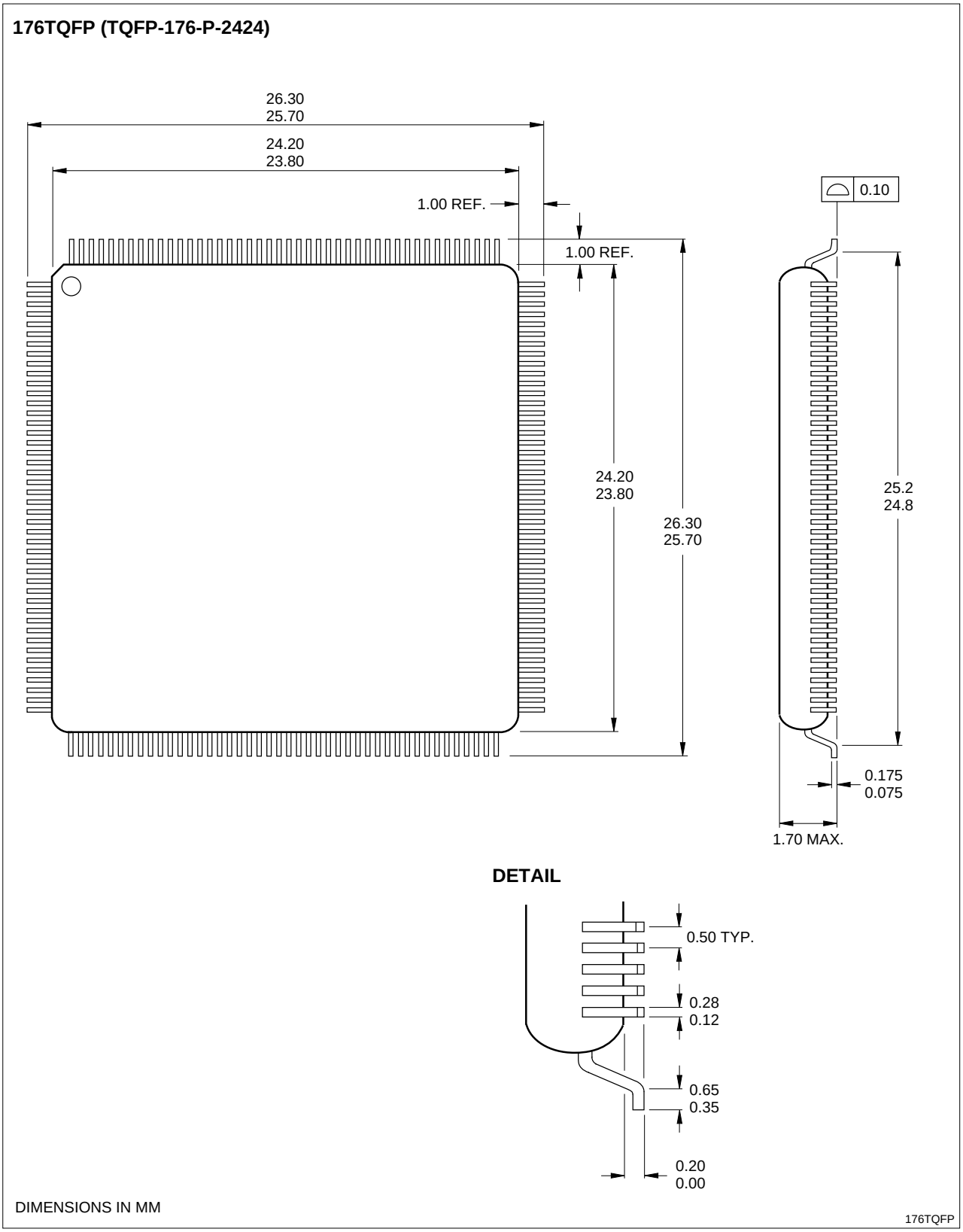


Figure 20-2. LH77790A Package Specification.

Chapter 21

REPRESENTATIVES & DISBRIBUTORS

REPRESENTATIVES

ALABAMA

Interep Associates
Daphne, AL
Tel: 334-621-1036
Huntsville, AL
Tel: 205-881-1096

ARIZONA

Electronic Tech. Sales
Tempe, AZ
Tel: 602-921-7122
Tucson, AZ
Tel: 602-883-3728

CALIFORNIA

Criterion Sales
Citrus Heights, CA
Tel: 916-725-0166
Santa Clara, CA
Tel: 408-988-6300

Jones & McGeoy Sales, Inc.
Newport Beach, CA
Tel: 714-724-8080

Mesa Components
San Diego, CA
Tel: 61-278-8021

COLORADO

Seale & Associates
Englewood, CO
Tel: 303-779-8005

CONNECTICUT

Datcom
Hamden, CT
Tel: 203-288-7005

FLORIDA

Gallagher & Associates, Inc.
Plantation, FL
Tel: 954-473-2101

Sun Marketing
Clearwater, FL
Tel: 813-536-5771

Sun Marketing
Deerfield, FL
Tel: 954-429-1077

Sun Marketing
West Melbourne, FL
Tel: 407-723-0501

GEORGIA

Interep Associates
Norcross, GA
Tel: 770-449-8680

ILLINOIS

Synergistic Sales
Glendale Heights, IL
Tel: 630-858-8686

INDIANA

Micro-Components
Kokomo, IN
Tel: 317-455-5000

IOWA

Spectrum Sales
Cedar Rapids, IA
Tel: 319-366-0576

KANSAS

Spectrum Sales
Prairie Village, KS
Tel: 913-648-6811

MARYLAND

Advanced Tech. Sales
Hunt Valley, MD
Tel: 410-771-0880

MASSACHUSETTS

Genesis Associates
Burlington, MA
Tel: 617-270-9540

MICHIGAN

Trilogy Marketing, Inc.
Auburn Hills MI
Tel: 810-377-4900
Portage, MI
Tel: 616-324-0049

MINNESOTA

The Twist Co.
Minneapolis, MN
Tel: 612-331-1212

MISSOURI

Spectrum Sales
St. Louis, MO
Tel: 314-731-4477

NEW JERSEY

Technical Marketing Group
West Caldwell, NJ
Tel: 201-226-3300

BGR-Wyck, Inc.
Mt. Laurel, NJ
Tel: 609-727-1070

Mycron Sales
Cranford, NJ
Tel: 908-272-7900

NEW YORK

Micro-Tech Marketing
Rochester, NY
Tel: 716-426-0806

Technical Marketing Group
Melville, NY
Tel: 516-351-8833

NORTH CAROLINA

EnVision, Inc.
Raleigh, NC
Tel: 919-878-3080

OHIO

Five Star Electronics
Dayton, OH
Tel: 937-299-1718
Hilliard, OH
Tel: 614-529-8031
Solon, OH
Tel: 216-349-1611

OKLAHOMA

RAM Electronics
Tulsa, OK
Tel: 918-491-6010

OREGON

BHC, Incorporated
Beaverton, OR
Tel: 503-641-1875

PENNSYLVANIA

Five Star Electronics
Pittsburgh, PA
Tel: 412-488-7352

SOUTH CAROLINA

EnVision, Inc.
Columbia, SC
Tel: 803-699-3360

TENNESSEE

Interep Associates
Greenville, TN
Tel: 615-639-3491

TEXAS

RAM Electronics
Austin, TX
Tel: 512-250-5082
El Paso, TX
Tel: 915-533-9707
Addison, TX
Tel: 214-713-4600
Houston, TX
Tel: 713-580-7394

VIRGINIA

Advanced Tech. Sales
Midlothian, VA
Tel: 804-378-5275

WASHINGTON

BHC, Incorporated
Lynnwood, WA
Tel: 206-774-8151

WISCONSIN

Synergistic Sales
Waukesha, WI
Tel: 414-544-5412

CANADA

Enerlec Sales Ltd.
Calgary, AB
Tel: 403-777-1550
Richmond, BC
Tel: 604-273-0882Indigo Electronics Ltd.
Kanata, ON
Tel: 613-591-1053
Mississauga, ON
Tel: 905-612-0025
Dollard desOrmeaux, PQ
Tel: 514-696-6104

ISRAEL

Ralco Components
Tel-Aviv
Tel: 972-3-6928233

MEXICO

Ciber Electronica
Guadalajara
Tel: 647-5217
Portales
Tel: 539-7832

PUERTO RICO

Sun Marketing Group
Caguas, PR
Tel: 787-745-1010

DISTRIBUTORS

ALABAMA

Marshall Industries
Huntsvillle, AL
Tel: 205-881-9235

Milgray Electronics
Madison, AL
Tel: 205-464-8646

Reptron Electronics
Huntsville, AL
Tel: 205-722-9500

Wyle
Huntsville, AL
Tel: 205-830-1119

ARIZONA

Marshall Industries
Phoenix, AZ
Tel: 602-496-0290

Sterling Electronics
Phoenix, AZ
Tel: 602-437-5565

Wyle
Phoenix, AZ
Tel: 602-804-7000

CALIFORNIA

Marshall Industries
Thousand Oaks
Tel: 805-370-5100
El Monte, CA
Tel: 800-522-0084
Irvine, CA
Tel: 714-458-5301
Milpitas, CA
Tel: 408-942-4600
Rancho Cordova, CA
Tel: 916-635-9700
San Diego, CA
Tel: 619-627-4140

Milgray Electronics
Irvine, CA
Tel: 714-753-1282
Milpitas, CA
Tel: 408-942-1600
San Jose, CA
Tel: 408-456-0900
San Diego, CA
Tel: 619-457-7545
Thousand Oaks, CA
Tel: 805-371-9399

Reptron Electronics
Irvine, CA
Tel: 714-450-0300
San Diego, CA
Tel: 619-453-8430
San Jose, CA
Tel: 408-453-9933

Sterling Electronics
Irvine, CA
Tel: 714-453-7660
San Diego, CA
Tel: 619-560-8097
San Jose, CA
Tel: 408-435-5566
Westlake Village, CA
Tel: 818-865-2333

Wyle
Calabasas, CA
Tel: 818-880-9000
Irvine, CA
Tel: 714-789-9953
Rancho Cordova, CA
Tel: 916-638-5282
Santa Clara, CA
Tel: 408-727-2500
San Diego, CA
Tel: 619-565-9171

COLORADO

Marshall Industries
Thornton, CO
Tel: 303-451-8383

Milgray Electronics
Westminster, CO
Tel: 303-657-2750

Sterling Electronics
Englewood, CO
Tel: 303-792-3939

Wyle
Thornton, CO
Tel: 303-457-9953

CONNECTICUT

Marshall Industries
Wallingford, CT
Tel: 203-265-3822

Milgray Electronics
Milford, CT
Tel: 203-878-5538

Reptron Electronics
Wallingford, CT
Tel: 203-265-3134

Sterling Electronics
Wallingford, CT
Tel: 203-265-9535

FLORIDA

Marshall Industries
Alamonte Springs, FL
Tel: 407-767-8585
Ft. Lauderdale, FL
Tel: 305-977-4880
St. Petersburg, FL
Tel: 813-573-1399

Milgray Electronics
Lake Mary, FL
Tel: 407-321-2555

Reptron Electronics
Ft. Lauderdale, FL
Tel: 305-735-1112
Tampa, FL
Tel: 800-659-1361

Sterling Electronics
Alamonte Springs, FL
Tel: 407-260-8558

Wyle
Deerfield Beach, FL
Tel: 305-420-0500
St. Petersburg, FL
Tel: 813-576-3004

GEORGIA

Marshall Industries
Norcross, GA
Tel: 770-923-5750

Milgray Electronics
Norcross, GA
Tel: 770-446-9777

Reptron Electronics
Norcross, GA
Tel: 770-446-1300

Sterling Electronics
Norcross, GA
Tel: 770-441-0449

Wyle
Norcross, GA
Tel: 770-441-9045

ILLINOIS

Marshall Industries
Schaumburg, IL
Tel: 847-490-0155

Milgray Electronics
Palatine, IL
Tel: 847-202-1900

ILLINOIS (cont'd)

Reptron Electronics
Schaumburg, IL
Tel: 847-882-1700

Sterling Electronics
Schaumburg, IL
Tel: 708-303-9900

Wyle

Addison, IL
Tel: 708-620-0969

INDIANA

Marshall Industries
Indianapolis, IN
Tel: 317-388-9069

Milgray Electronics
Indianapolis, IN
Tel: 317-781-9997

KANSAS

Marshall Industries
Lenexa, KS
Tel: 913-492-3121

Milgray Electronics
Overland Park, KS
Tel: 913-236-8800

Sterling Electronics
Lenexa, KS
Tel: 913-492-5406

MASSACHUSETTS

Marshall Industries
Wilmington, MA
Tel: 508-658-0810

Milgray Electronics
Wilmington, MA
Tel: 508-687-5900

Reptron Electronics
Burlington, MA
Tel: 617-273-2800

Sterling Electronics
Woburn, MA
Tel: 617-938-6200

Wyle
Bedford, MA
Tel: 617-271-9953
Burlington, MA
Tel: 617-272-7300

MARYLAND

Marshall Industries
Columbia, MD
Tel: 410-880-3030

Milgray Electronics
Columbia, MD
Tel: 410-730-6119

Reptron Electronics
Columbia, MD
Tel: 800-669-0694

Sterling Electronics
Columbia, MD
Tel: 410-290-3800

Wyle
Columbia, MD
Tel: 410-312-4844

MICHIGAN

Marshall Industries
Livonia, MI
Tel: 313-525-5850

Reptron Electronics
Livonia, MI
Tel: 313-525-2700

MINNESOTA

Marshall Industries
Plymouth, MN
Tel: 612-559-2211

Reptron Electronics
Plymouth, MN
Tel: 612-559-0000

Sterling Electronics
Eden Prairie, MN
Tel: 612-946-8888

Wyle
Bloomington, MN
Tel: 612-853-2280

MISSOURI

Marshall Industries
Earth City, MO
Tel: 314-770-1749

NEW JERSEY

Marshall Industries
Fairfiled, NJ
Tel: 201-882-0320
Mt. Laurel, NJ
Tel: 609-234-9100

Milgray Electronics
Mt. Laurel, NJ
Tel: 800-257-7111
Parsippany, NJ
Tel: 201-335-1766

Sterling Electronics
Parsippany, NJ
Tel: 201-331-0999
Mt. Laurel, NJ
Tel: 609-273-6420

Wyle
Mt. Laurel, NJ
Tel: 609-439-9110
Oradill, NJ
Tel: 201-261-3200
Pine Brook, NJ
Tel: 201-882-8358

NEW MEXICO

Sterling Electronics
Albuquerque, MN
Tel: 505-884-1900

NEW YORK

Marshall Industries
Endicott, NY
Tel: 607-785-2345
Rochester, NY
Tel: 716-235-7620

Milgray Electronics
Farmingdale, NY
Tel: 800-645-4729
Pittsford, NY
Tel: 716-381-9700

Reptron Electronics
Hauppauge, NY
Tel: 516-952-3196

NORTH CAROLINA

Marshall Industries
Raleigh, NC
Tel: 919-878-9882

Milgray Electronics
Raleigh, NC
Tel: 919-790-8094

Reptron Electronics
Raleigh, NC
Tel: 919-870-5189

Sterling Electronics
Raleigh, NC
Tel: 919-790-8634

Wyle
Morrisville, NC
Tel: 919-469-1502

OHIO

Marshall Industries
Dayton, OH
Tel: 937-898-4480
Solon, OH
Tel: 216-248-1788

Milgray Electronics
Cleveland, OH
Tel: 216-447-1520

OHIO (cont'd)

Reptron Electronics
Solon, OH
Tel: 216-349-1415

Sterling Electronics
Solon, OH
Tel: 216-248-1122

Wyle
Solon, OH
Tel: 800-763-9953
216-248-9996

OKLAHOMA

Sterling Electronics
Tulsa, OK
Tel: 918-663-2410

OREGON

Marshall Industries

Beaverton, OR
Tel: 503-644-5050

Milgray Electronics
Beaverton, OR
Tel: 800-986-2299

Reptron Electronics
Beaverton, OR
Tel: 503-629-2082

Sterling Electronics
Beaverton, OR
Tel: 503-643-9090

Wyle
Beaverton, OR
Tel: 503-643-7900

PENNSYLVANIA

Reptron Electronics
Horsham, PA
Tel: 800-251-2617

TEXAS

Marshall Industries
Austin, TX
Tel: 512-837-1991
Houston, TX
Tel: 713-467-1666
Richardson, TX
Tel: 214-705-0600

Milgray Electronics
Austin, TX
Tel: 512-331-9961
Dallas, TX
Tel: 972-248-1603
Houston, TX
Tel: 713-870-8102

Sterling Electronics
Austin, TX
Tel: 512-836-1341
Carrollton, TX
Tel: 214-243-1600
Houston, TX
Tel: 800-745-5500

Wyle
Austin, TX
Tel: 512-833-9953
Houston, TX
Tel: 713-784-9953
Richardson, TX
Tel: 214-235-9953

UTAH

Marshall Industries
Salt Lake City, UT
Tel: 801-973-2288

Milgray Electronics
Murray, UT
Tel: 801-261-2999

Sterling Electronics
Salt Lake City, UT
Tel: 801-972-5444

Wyle
Orem, UT
Tel: 800-414-4144
801-226-0991
West Valley City, UT
Tel: 801-974-9953

VIRGINIA

Sterling Electronics
Richmond, VA
Tel: 804-226-2190

WASHINGTON

Marshall Industries
Bothell, WA
Tel: 206-486-5747

Reptron Electronics
Redmond, WA
Tel: 206-702-9166

Sterling Electronics
Kirkland, WA
Tel: 206-823-9770

Wyle
Redmond, WA
Tel: 206-881-1150

WISCONSIN

Marshall Industries
Waukesha, WI
Tel: 414-797-8400

Wyle
Brookfield, WI
Tel: 414-879-0434

CANADA

Future Electronics
Calgary, AB
Tel: 403-250-5550
Edmonton, AB
Tel: 403-438-2858
Mississauga, ON
Tel: 416-612-9200
Ottawa, ON
Tel: 727-1800
Pointe Claire, PQ
Tel: 514-694-7710
Quebec City, PQ
Tel: 418-877-6666
Vancouver, BC
Tel: 604-294-1166
Winnipeg, MB
Tel: 204-786-7711

Marshall Industries
Calgary, AB
Tel: 403-274-3717
Mississauga, ON
Tel: 905-612-1771
Pointe Claire, PQ
Tel: 514-694-8142
Vancouver, BC
Tel: 604-294-6506

Milgray Electronics
Burnaby, BC
Tel: 604-291-0044
Mississauga, ON
Tel: 905-678-0958
Pointe Claire
Tel: 514-426-5900
Vancouver, BC
Tel: 604-291-0044

SEMAD/Sterling Electronics
Burnaby, BC
Tel: 604-451-3444
Calgary, AB
Tel: 403-252-5664
Markham, ON
Tel: 905-475-3922
Ottawa, ON
Tel: 613-526-4866
Pointe Claire, PQ
Tel: 514-694-0860

PUERTO RICO

Milgray Electronics
Canovanas, PR
Tel: 809-876-8200

SHARP

NORTH AMERICA

SHARP Electronics Corporation
Microelectronics Group
5700 NW Pacific Rim Blvd., M/S 20
Camas, WA 98607, U.S.A.
Phone: (360) 834-2500
Facsimile: (360) 834-8903
Telex: 49608472 (SHARPCAM)
http://www.sharpmeg.com

EUROPE

SHARP Electronics (Europe) GmbH
Microelectronics Division
Sonninstraße 3
20097 Hamburg, Germany
Phone: (49) 40 2376-2286
Facsimile: (49) 40 2376-2232
Telex: 2161867 (HEEG D)

ASIA

SHARP Corporation
Integrated Circuits Group
2613-1, Ichinomoto-Cho
Tenri-City, Nara, 632, Japan
Phone: (07436) 5-1321
Facsimile: (07436) 5-1532
Telex: LABOMETA-B J63428

Reference No.: SMT96100-B