

# Inside SQ80 – a technical description

Rainer Buchty

eMail: [rainer@buchty.net](mailto:rainer@buchty.net)

<http://www.buchty.net>

September 29, 1999



### **Abstract**

Since Ensoniq refuses to give any technical information about the SQ80 Cross Wave Synthesizer someone had to do the job... I hope this documentation will help the electronically skilled to repair and/or improve their beloved machine. At least it will give some deeper understanding of what's going on inside.

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	What is the (E)SQ family of synthesizers all about? . . . . .	6
1.2	Synthesis Parameters . . . . .	6
1.3	The sequencer . . . . .	8
1.4	Multitimbrality - I need more voices! . . . . .	8
<b>2</b>	<b>Technical Documentation</b>	<b>11</b>
2.1	System Overview . . . . .	12
2.2	The one which drives it all: MC6809E . . . . .	13
2.3	Let there be sound: DOC5503 and CEM3379 . . . . .	14
2.4	The art of disk storage . . . . .	15
2.4.1	Track Layout . . . . .	15
2.4.2	Disk Directory Structure . . . . .	16
2.4.3	Sequencer Memory Dumps . . . . .	16
2.4.4	Bank Files . . . . .	17
2.4.5	Program Files . . . . .	17
2.5	What comes in, must come out - the I/O subsystem . . . . .	19
2.6	The system software . . . . .	20
2.6.1	OSRAM . . . . .	20
2.6.2	Cartridge . . . . .	20
2.6.3	Hidden Functions . . . . .	20
<b>3</b>	<b>Troubleshooting</b>	<b>23</b>
3.1	General problems . . . . .	24
3.2	MIDI mysteries . . . . .	25
3.3	Keyboard Trouble . . . . .	25
3.4	Storage Hassles . . . . .	26
3.5	Panel Problems . . . . .	26
3.6	Flaky Tape . . . . .	26
<b>A</b>	<b>Parts List</b>	<b>27</b>
<b>B</b>	<b>Schematics</b>	<b>29</b>

<b>C Datasheets</b>	<b>39</b>
C.1 WD1770/1772 Floppy Disk Controller / Formatter . . . . .	41
C.1.1 Description . . . . .	41
C.1.2 Architecture . . . . .	41
C.1.3 Processor Interface . . . . .	43
C.1.4 General Disk Read Operation . . . . .	45
C.1.5 General Disk Write Operation . . . . .	45
C.1.6 Command Description . . . . .	46
C.1.7 Type 1 Commands . . . . .	48
C.1.8 Type 2 Commands . . . . .	50
C.1.9 Type 3 Commands . . . . .	52
C.1.10 Type 4 Commands . . . . .	54
C.1.11 Status Register . . . . .	55
C.1.12 Recommended Layout for 128-Byte Sectors . . . . .	56
C.1.13 Recommended Layout for 256-Byte Sectors . . . . .	57
C.1.14 Generic (non-standard) formats . . . . .	58
C.2 SSM2300 Octal Sample&Hold . . . . .	61
C.3 MC/SN 2681 DUART . . . . .	63
C.4 CEM3360 Dual VCA . . . . .	89
C.5 CEM3379 Analog Signal Processor . . . . .	93
C.6 Ensoniq DOC5503 . . . . .	103
C.6.1 Common Registers . . . . .	103
C.6.2 DOC registers for individual Oscillators . . . . .	103
C.6.3 Wavetable Address Generation . . . . .	106
C.6.4 Pinout . . . . .	107
C.7 MC/HD 68B09E CPU . . . . .	109
C.8 AD7524 Analog/Digital Converter . . . . .	145



# Chapter 1

## Introduction

The SQ80 was released at the beginning of 1988 (at least here Europe) and was one of the first so-called workstations. It not only was a synthesizer but also came along with a 8-track sequencer. Obviously, it also featured multi-timbrality and dynamic voice allocation which was quite a novum. And as one of the first synthesizers it was equipped with a 3.5" disk drive - no need for expensive sound cartridges or excessive bulk dump sessions anymore.

What also made the (E)SQ family of synthesizers a success was the easy-to-use user interface. Unlike most synthesizers of that time where one had to select a parameter by number (!) and editing that one again by number (sometimes even in hexadecimal) the SQ80 came (as its little brother the ESQ-1) with a big alpha-numeric display and a load of buttons each being responsible for a group of functions.

Unfortunately, the early Ensoniq synths and accessories for these such as the single output expansion unit disappeared completely after the first TransWave<sup>TM</sup> synthesizer – the VFX – hit the market.

## 1.1 What is the (E)SQ family of synthesizers all about?

The (E)SQ synthesizers as well as Ensoniq's first sampler (which was a great success) are based on the so called DOC chip. This abbreviation stands for **D**igital **O**scillator **C**ircuit and is the musical heart of all of these machines. Speaking of the synthesizers we could say that the SQ80 is a superset of the ESQ-1 offering the following enhancements:

- bigger waveform memory (75 waves including the 32 ESQ-1 waves)
- bigger sequencer memory (64kB by default)
- built-in 3.5" disk drive
- keyboard with polyphonic aftertouch

What makes the (E)SQ synthesizers (even today!) quite interesting is their great flexibility. They can either sound warm and analog but also cold and digital. The reason lies in their hybrid nature – digital oscillators but analog sound processing. The factory presets give a rough overview over the SQ80's capabilities but to get a real impression of what this machine can do should get e.g. the Transoniq Hacker patches.

## 1.2 Synthesis Parameters

Figure 1.1 shows the basic architecture of each voice:

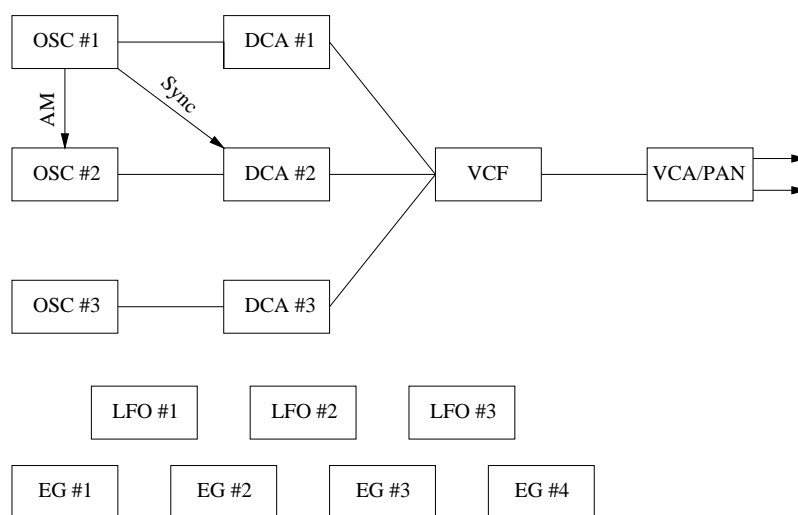


Figure 1.1: Architecture of an SQ80 voice



Unlike modern synthesizers where a single voice mostly consists of an oscillator plus filters the SQ80 offers 3 oscillators per voice (using 4 oscillators might have overextended the CPU's capabilities, I just don't know) which gives it a rich base sound. Also pretty nice is that per voice four envelope generators and three LFOs exist – soundprogrammer's heaven. Besides, the SQ80 offers some more features which I will explain as follows:

**Gated Mode:** Besides some one-shot (not loopable) waveforms the SQ80 offers a special playback mode where an Envelope is only processed once treating the sustain level as just an ordinary envelope step. This is nice for producing percussive sounds which don't need a sustain phase.

**Synchronization:** You know that fancy “EEEEOOOW” sounds? It's the audible effect of one oscillator synchronizing another. The SQ80 can produce these, too, since the DOC chip supports oscillator synchronization - unfortunately only between an even/odd pair of oscillators, that's why only synchronizing OSC2 by OSC1 is possible.

**Amplitude Modulation:** Good for making gong sounds or any other which need disharmonic spectrals. Not quite the same as ring modulation but pretty close.

**Oscillator Restart** In usual DCO-based synthesizers the oscillator starts playing back a wave from its very beginning when a key is played. This is ok for complex waves but results in a static sound when using short, looped waves such as e.g. SAW. Thinking of analog oscillators such a reset of the waveform is unnatural (saw oscillators are based on integrators) - the SQ80 offers both: oscillator reset or free-running waves. However, this can be only programmed per sound program, not individual for each oscillator.

**LFO Specialties:** Instead of a fixed output amplitude the modulation depth of each LFO can be programmed to fade in (or out). If that's not enough an assignable modulation depth modulator exists.

Digitally controlled synthesizers tend to offer very static LFOs: As soon as you hit a key the LFO starts at position 0 of the selected wave - this is very annoying when using the LFO for e.g. filter sweeps, so the SQ80's LFOs can be programmed to be running freely or being reset by each keystroke. And if you think that digital LFOs sound too static – switch on the HUMANIZE function!

**Envelope Specialties:** Not much uncommon here, but for completeness' sake I'd mention that an envelope generator can be programmed

to react on key velocity (linear and exponential response) and key position. What's really fancy about the envelopes is that the so called *second release* which does not really replace a reverb but produces a similar effect.

Of course the envelopes are not of ADSR-type but a 4 level/rate model.

### 1.3 The sequencer

Normally, on-board sequencers are a nice add-on but nothing more. Not the SQ80 built-in sequencer which is really useful. It's a pattern-oriented sequencer capable of holding up to 20 songs consisting of up to 60 sequences. What makes it so useful is the fact that you not only have a basic record/quantize/playback functionality but also the capabilities of track transpose, sequence editing (add/delete bars) and even single step editing of track contents. Also it contains a very flexible song editor and locator. Needless to say that the sequencer not only can be controlled by MIDI but also synchronized to MIDI clock or Tape Sync.

On the "minus" side there's only few. Have you ever tried to attach a new volume level to an existing track after having recorded it? On earlier software versions this is impossible.

### 1.4 Multitimbrality - I need more voices!

The SQ80 was one of the first synthesizers offering multitimbrality which means that one machine was able to play *independent* sounds on each voice (to that time we used the term *MIDI multi mode capable* which is pretty the same).

Unfortunately, the SQ80 offers only 8 voices which is not much if you think of complex arrangements. Fortunately, it has a really neat voice allocation algorithm (much better as the very static one used e.g. in the Yamaha SY-77) which can be influenced per sound program individually. Think of natural instruments - if you play the same note twice the previously played note will be replaced by the following one. The SQ80 can mimic this behaviour - or disregard nature, just as you like. But keep in mind that the latter one forces a more aggressive voice allocation since there's no oscillator to "recycle".

Lucky people (such as me :-)) own more than one SQ80 - and can easily daisy-chain these by enabling the *overflow mode*. This means that any note

which an SQ80 can't play since all oscillators are used will be handed down the chain to another SQ80 which might have free resources.



## Chapter 2

# Technical Documentation

I've asked Ensoniq three times for the original technical documentation of the SQ80, here's what they responded:

1. *"The schematics are proprietary and not meant to be given away."* (This was after my naive(?) first-time asking, but hey, I got the schematics of my Yamaha equipment aswell - and these are far more high-tech!)
2. *"Your work sounds incredible – but we wish you all the best."* (after telling them that I started to reverse-engineer the SQ80)
3. *"Congratulations on your work!"* (when finally telling them that I succeeded in drawing schematics, documenting the OS and creating an assembly language source code for further OS development)

With no word they told me that they dislike these efforts, neither they told me to keep that knowledge for myself. So here it comes...

## 2.1 System Overview

Basically, the SQ80 is nothing more than a microcomputer with some specialized peripherals. Figure 2.1 gives an overview over the system architecture:

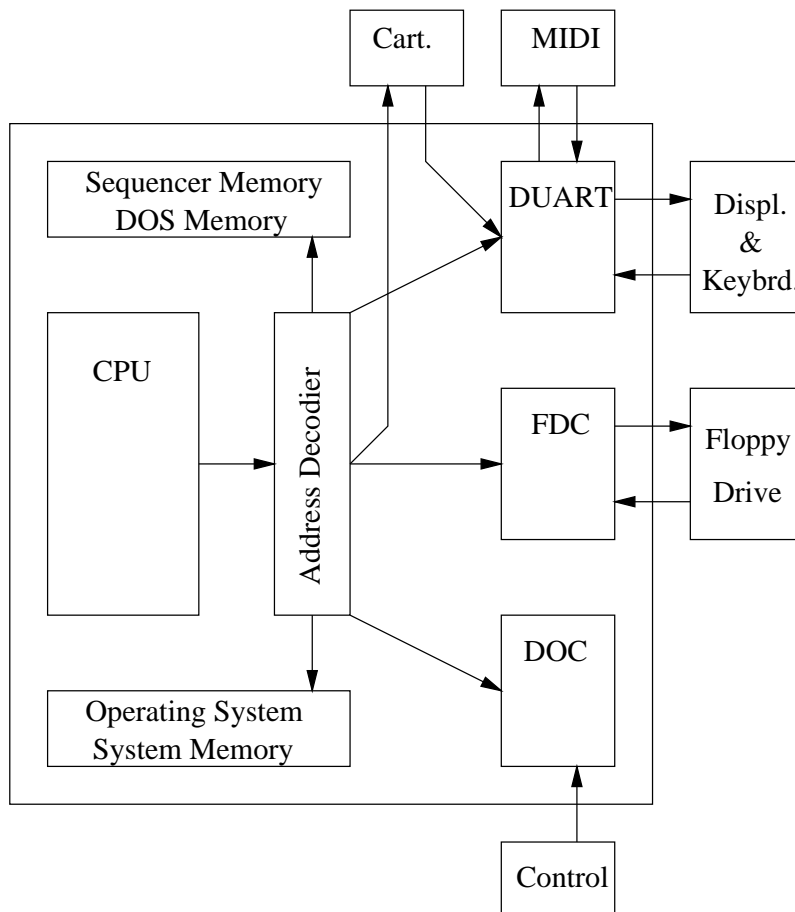


Figure 2.1: Anatomy of an SQ80

Mostly computer stuff, the only dedicated “musical” parts are the DOC and the analog sound processors. This is a big advantage over other, especially modern synthesizers, since you can get replacement parts from the next distributor of electronic parts.

## 2.2 The one which drives it all: MC6809E

The Motorola 6809 was widely used for synthesizers back in the 80's. Not only Ensoniq did use it in their (E)SQ synthesizers and the Mirage samplers, it's also the heart of e.g. the Oberheim XPander. And even the DX-7 used some special kind of 6809, the Hitachi 6309 (an improved 6809).

What made the 6809 that attractive was its computing power combined with a very flexible instruction set. The SQ80 makes heavy use of the 6809's specialties:

- loadable system stack pointer S (used for context switching)
- loadable user stack pointer U (used as another 16-bit index register)
- 16-bit index registers X and Y
- 16-bit accumulator D
- fast (integer) multiplication / division
- software interrupts (used for error signalling)
- SYNC and CWAI commands forcing the CPU to wait for an external interrupt

In the SQ80 the 6809E was used. Unlike the 6809 it's E-mate could be completely externally controlled which is needed inside the SQ80 where the generation of clock signals is influenced by the DOC chip. See p.111 for details on this processor.

### 2.3 Let there be sound: DOC5503 and CEM3379

Some of you might know that the creator of the DOC, Bob Yannes, was also the designer of the all famous SID chip which was the sound engine of the Commodore 64 home computer. Both are based on the same principle: phase accumulation. This means that the frequency of a digital voice is not determined by the playback frequency (clock) as it was used in former drum computers (and which will drive every today's studio technician mad since a variable playback frequency won't fit very well into digital mastering using fixed frequencies of 44.1kHz or 96kHz), instead the frequency will be derived from a counter: This counter – roughly said – counts up the waveform address. If you need to get higher frequencies you increment the counter steps, for lower frequencies you decrement them. Thus the name “phase accumulator” – it's an accumulator where the resulting address is a phase pointer.

The DOC uses a 24bit counter for this purpose. The frequency-determining value will be added to the lower 16 bit whereas the higher bits are used as the phase pointer (which of them will be used can be programmed, see 107 for details).

The DOC contains 32 digital oscillators and 32 amplifiers aswell as an output multiplexer to make it possible to route these 32 oscillators to one out of 16 channels. The SQ80 makes only use of 24 oscillators/amplifiers and 8 channels. The uppermost channel mux bit is used for selecting either Wave ROM 0 or 1.

Now the CEM3379: It was designed by Curtis Electromusic and contains an analog 4pole low-pass filter with adjustable center frequency and resonance. Furthermore it also contains a dual VCA with adjustable gain and pan position. Each SQ80 voice contains mainly of one CEM3379, the DOC's output is routed to the desired CEM3379 using an ordinary 4051 multiplexer.



## 2.4 The art of disk storage

The SQ80 uses standard 3.5" DD disks for storage of programs, program banks and sequencer data. As disk controller it uses the Western Digital WD1772 which was quite common in the 80s, the same chip is used e.g. inside the Atari ST machines and the Commodore 1581 disk drive.

The disks have a fixed geometry: Although one disk may hold up to 880kB of data it's not possible to make dynamic use of a disk in a way of storing e.g. up to 220 program banks or up to 8834 single programs. Neither it is possible to store more than 10 sequencer files even if they don't make use of the entire disk. Instead the fixed disk geometry causes the well known layout of 10 sequencer/sysex files, 40 program banks and 128 single programs – which unfortunately most of the time is just wasting disk space.

### 2.4.1 Track Layout

Some of you might have tested it: It's almost impossible to copy a SQ80 disk using standard PC drives. This is due to the fact that Ensoniq – in their eternal wisdom – decided to use a very special disk format. Not only that they use a fixed disk layout (which is understandable in terms of complexity: A fixed disk layout is just way easier to program and needs less administration – on the other hand it wouldn't have been a big problem to modify the SQ80 hardware to have additional ROM space which would be needed for the implementation of a “real” DOS making dynamic use of the disk space), they also use a special track/sector layout:

- sectors 0 to 4 hold 1024 data bytes each
- sector 5 holds 512 data bytes

The reason why a lot of PC disk controllers fail to read and even write these disks is the change in sector size inside a track. Whereas the WD177x family can perform single sector reads and writes older PC floppy controllers such as the NEC 765 or Intel 8278 based ones can only read and write a single track. More modern ones like Intel 82077 and newer are at least able to read and write single sectors – but not format these individually.

Have you ever wondered why the SQ80 won't format a disk where your PC formats it without bad sectors? That's because Ensoniq formats DD disks to their limit (880kB / 901120 bytes) – which also made the Commodore Amiga be very sensitive to cheapo disks. (Mean trick: Let the SQ80 format and complain, then put the disk into your PC and copy disk images using the SQ80 Toolkit onto it.

### 2.4.2 Disk Directory Structure

The disk directory consists of two parts: The first part is responsible for “big files”, namely sequencer memory dumps and sound banks. Thus, it holds 50 entries of the following structure:

```
typedef struct directory_entry {
    char type;
    char name[10];
    int size;
} de_t;
```

Size and type are only vital for sequencer files since they tell, guess what... File size is measured in bytes, but the file type needs some more explanation:

```
/* standard file types */
#define FREE    0    /* unused */
#define OS      1    /* operating system */
#define BNK     2    /* program bank */
#define SNG     3    /* all sequence (song) */
#define SEQ     4    /* one sequence */
#define SYX     5    /* system exclusive */
#define PRG     6    /* single program */
```

I guess you get it from the above table. The very first entry is used for unused or deleted file, the second one is not used on the SQ80 and is a remnant of the good old Mirage times - it’s reserved for bootable system files.

But what about the single program files? It’s definitely not true that you need to extract the program names from a program files PCB structure - if that would be the case you’d hear annoying floppy noises each time you access the DISK/LOAD/PROGRAM menu. Instead, the file names are listed just behind that 50 “big” entries.

### 2.4.3 Sequencer Memory Dumps

This kind of files occupies the first and biggest disk partition, you will find these at the following locations (format: start c/h/s, end c/h/s):

```
{ 0,0,0, 6,0,3},    /* seqram #1 */
{ 6,0,4,12,1,2},   /* seqram #2 */
{12,0x11,3,19,1,1}, /* seqram #3 */
{19,1,2,25,0,0},   /* seqram #4 */
{25,0x10,1,31,0,4}, /* seqram #5 */
```

```

{32,0,0,38,0,3},      /* seqram #6 */
{38,0,4,44,1,2},     /* seqram #7 */
{44,0x11,3,51,1,1},  /* seqram #8 */
{57,1,2,57,0,0},     /* seqram #9 */
{57,0x10,1,63,0,4},  /* seqram #10 */

```

Looks easy? It indeed is - but not as simple as it looks like. Your SQ80 is a lazy guy which changes disk sides only when it's necessary. This means a multi-track read goes like this:  $c/0 - c/1 - c+1/1 - c+1/0 - c+2/0$  and so on. To mark whether the head has changed before or not bit 4 of the head byte is used - if it is set to 1 it forces a head change together with the next track change.

But there's even more: On sector 5 of the end track/side there's additional information stored such as song names. You'll see the side effects of this when discussing the storage of single program files.

#### 2.4.4 Bank Files

Program bank file storage is easy: Each bank occupies 4 long sectors on disk starting with track 64 and ending with track 79. Banks 1 to 20 will be stored on side 0, banks 21 to 40 on side 1 of a disk. Each bank consist of 40 single program dumps organized as PCBs.

In the directory structure these files are marked with file type 2 (BNK) and size 0.

#### 2.4.5 Program Files

Getting a program file's position on disk is a tricky task. In theory the single programs reside on the remaining (read: not occupied by the directory) sectors 5 of a disk - but remember what I said about the sequencer files: These need another 10 short sectors which leads to the following routine:

```

geo_t *get_prog_pos(geo_t *prgpos, int pnum) {
    switch( ((pnum-1)&64)|(((pnum-1)&63)+2) ) {
        case 0x06:      prgpos->st=0x42;
                        prgpos->sh=0; break;
        case 0x19:      prgpos->st=0x42;
                        prgpos->sh=1; break;
        case 0x1f:      prgpos->st=0x43;
                        prgpos->sh=1; break;
        case 0x39:      prgpos->st=0x44;
                        prgpos->sh=0; break;
        case 0x3f:      prgpos->st=0x44;
    }
}

```

```

                                prgpos->sh=1; break;
case 0x53:                       prgpos->st=0x45;
                                prgpos->sh=1; break;
case 0x6c:                       prgpos->st=0x46;
                                prgpos->sh=0; break;
case 0x73:                       prgpos->st=0x46;
                                prgpos->sh=1; break;
default:                         prgpos->st=((pnum-1)&63)+2;
                                prgpos->sh=((pnum-1)&64)>>6;
                                break;
}
prgpos->ss=5;
prgpos->et=prgpos->st; prgpos->eh=prgpos->sh; prgpos->es=prgpos->ss;
return prgpos;
}
```

Needless to say that the storage format – again – is PCB.

## 2.5 What comes in, must come out - the I/O sub-system

Big words for a small chip - but indeed the MC2681 DUART is a vital part of any SQ80 since it's responsible for the following functions:

- MIDI communication (Serial Port A)
- communication with keyboard processor (Serial Port B)
- communication with panel processor (Serial Port B)
- tape I/O
- cartridge presence checking
- disk head access
- disk change detection
- selection of voltages to be sampled
- OSROM low bank switching
- SEQRAM selection and bank switching
- metronome click generation
- timing

To be more precise, the input and output lines are used for the following tasks:

Inport	Task	Outport	Task
0	Tape In	0	Disk Head
1	Disk Change Detection	1	Multiplexer Bit 0
2	Cartridge Detection	2	Multiplexer Bit 1
3	500Hz Interrupt	3	Multiplexer Bit 2
4	500Hz Interrupt	4	Metronome Click Generation
5	1kHz Interrupt	5	Metronome Click Generation
6	1kHz Interrupt	6	Tape Out
-		7	Tape Out

Table 2.1: Additional DUART tasks

## 2.6 The system software

Now this is something which really impressed me: The SQ80 has a really nice multitasking real-time operating system (RTOS). Normally, you don't get in contact with the mysteries of this OS, but just for completeness' sake I'll give a rough overview:

### 2.6.1 OSRAM

The OSRAM needs to be "formatted" since some vital data structures are stored here, the most relevant one is the context table which is responsible for proper task switching. Since the OSRAM is battery buffered (it also holds the sound programs) you normally don't run into problems, even when you need to replace the battery after some years (mine is doing fine for over 11 years now), since the OS checks for an empty OSRAM during reset and reformats it if necessary. When upgrading the OS ROMs this check might fail and you need to reformat it manually (see 2.6.3 for details).

### 2.6.2 Cartridge

You know the cartridge as something to hold program banks. Like the OSRAM it also holds some status data which is explained below:

**0x3FFD:** set to 0 if cartridge bank B contains data, 0xff else

**0x3FFE:** set to 0 if cartridge bank A contains data, 0xff else

**0x3FFF:** set to 0x01 if cartridge contains program banks, it's also possible to take over the system if 0x55 is stored here.

If a cartridge is present or not is detected by the DUART's input port 2.

### 2.6.3 Hidden Functions

Yes, the SQ80 offers hidden functions which are not mentioned in the Musician's Manual. You can reach these by pressing RECORD with one of the following keys:

#### **COMPARE**

Analog Voltage Check

#### **FILTER**

With this function the SQ80 recalibrates its filters. This is to ensure that all 8 voices have (nearly) the same filter response parameters - unfortunately, this tuning is responsible for the SQ80's filter not being able to self oscillate: The filter tuning parameters are calculated in a way that self-oscillation is just impossible.

**MASTER**

Prints the OS version. The latest version released by Ensoniq was 1.8, unfortunately there's no official support anymore but if you are able to program EPROMs yourself you'll find the images on my web page.

**MODES**

If you ever wondered who built the SQ80 call this menu.

**Soft Button 1**

Reinitialization ("OSRAM Formatting") - as mentioned above this function is needed after an OSROM upgrade.

**Soft Button 6**

Warm Reset. Nice idea, but jams the machine - at least on OS versions 1.7 and 1.8.

**SPLIT/LAYER**

Keyboard recalibration.





## Chapter 3

# Troubleshooting

Almost all eMails I get concerning the SQ80 are about the synthesizer behaving strange or not working at all. On the following pages you'll find hints about what might cause misbehaviour and how to fix it.

### 3.1 General problems

**The system won't come up...**

If your SQ80 does not come up at all, showing a blank display and no reaction to any MIDI message or key pressure its most likely the power supply. Check for blown fuses or dried-out capacitors. If that's not the problem replace U6 and/or U10.

**I've upgraded the OS and now the system doesn't come up!**

Just press RECORD together with soft button 1 (the upper left above the display). This will perform a complete reset including reformatting the OSRAM. Afterwards, everything should be ok again. If not check the OSROMs for correct placement.

**Display says that the battery voltage is low.**

Just replace it. Any 3V to 5V lithium battery will be fine. If you don't find the battery on the motherboard you should probably leave that step to somebody who knows on which side the soldering iron heats...

**The pedal won't work.**

Check cabling. Otherwise replace U33 – if that doesn't help either your DOC is f\*cked up, try to get one from a used/dead Apple IIGS, ESQ-1 or SQ80. Or send it to me so that I can make an expander version out of it.

**The wheels won't work.**

Check pedal cabling. Rest see above.

**My cartridge isn't recognized.**

Check if cartridge is formatted. Check cabling. Check presence of 5V at pin x of U6 while cartridge is inserted. Voltage found? Build a new cable using new connectors. If that won't help replace U6. If you did not find the voltage replace U2 to U4.

**There are no metronome clicks anymore.**

Replace U39. If that doesn't help replace U44. If that didn't help either replace U6.

**I can't hear anything.**

Check pins 6 and 9 of U44 if it gets any signals. If this is not the case replace U39, otherwise replace U44.

**Some voices are missing.**

Check pin 8 of U40-43, U45-48 playing an 8-note chord. If one or more doesn't get an input signal replace U36. If that doesn't help search for a new DOC and replace U27.

**The voices are stuck!**

Check sustain switch. Replace U33.

**The sustain switch doesn't work.**

See above.

**The pedal doesn't work.**

Check pedal. Replace U33.

**The Sequencer switch doesn't work.**

Check switch. Replace U33.

If they all get input signals identify the one which does not output a signal on pin 15 and 17 (be sure to have pan set to 8 with no modulation while testing this) and replace it.

## 3.2 MIDI mysteries

**I can't send MIDI messages, but receiving them is fine.**

Replace Q1 to Q4. If that doesn't help, replace U6.

**I can't receive MIDI messages, but sending them is fine.**

Replace U12. If that doesn't help, replace U6.

**I can only receive/send some MIDI messages.**

Duh! Go to MIDI menu and enable the desired messages.

## 3.3 Keyboard Trouble

**During reset display says keyboard is disabled!?**

Check cabling. Power the system on and check U6 pin 10 and 11 for serial communication. If you don't see anything at pin 10 replace U1 (68HC11) on the keyboard PCB – otherwise replace U6.

**I get keyboard processor errors.**

Check cabling. Replace U1 (68HC11) on the keyboard PCB. If that doesn't help replace U6.

**Keyboard calibration fails all the time.**

Check cabling. If that doesn't help try to get a new keyboard ASIC or send the SQ80 to me to make a nice expander version out of it.

**The SQ80 won't leave keyboard calibration.**

Check cabling. Check keyboard PCB for cold soldering spots. Check serial communication (see above).

**My keyboard doesn't work – but I get no error message!**

Duh! Go to MASTER menu and enable it.

### 3.4 Storage Hassles

**I get more and more read/write errors.**

Clean the drive using an ordinary cleaning diskette – or be tough and clean the drive heads using Q-tips and isopropanole.

If that won't help replace the drive. Any 720kB drive with shugart bus will do the job.

**The drive doesn't show any reaction.**

Check cabling. Replace U24 and U29. If that won't help, replace U9.

### 3.5 Panel Problems

**I can't see anything...**

Check cabling. Check for correct display voltage. If that's not the problem check/replace the display drivers. If that won't help, send the SQ80 to me to make a nice expander version from it since either the display itself or the panel CPU is broken. No replacement possible.

**Some keys won't work.**

Remove the panel PCB and clean the contacts.

**Some keys are stuck.**

Remove the panel and clean the keys / holes.

### 3.6 Flaky Tape

**I can't load from tape.**

Replace U25. If that doesn't help replace U6.

**I can't sync in.**

see above

**I can't write to tape.**

Replace U6.

**I can't sync out.**

see above

## Appendix A

### Parts List

#	Type	Description	Task
Q1	2N3906		MIDI out
Q2	2N3904		MIDI thru
Q3	2N3906		MIDI thru
Q4	2N3906		MIDI out
Q6	JE182		Amplification
Q7			Amplification
U1	74LS04	hex inverter	
U2	74ALS245	8x bidir. buffer/driver	Cartridge Data
U3	74ALS244	8x buffer/driver	Cartridge Address/Control
U4	74ALS244	8x buffer/driver	Cartridge Address/Control
U5	74LS161	4bit counter	Timer
U6	MC2681, SCN2681	DUART	Communication & Control
U7	74LS10	3x 3-input NAND	misc. Cartridge
U8	74LS00	4x 2-input NAND	Clock Generation
U9	WD1772	FDC	Disk Drive Control
U10	MC6809E	CPU	Central Processing Unit
U11	74LS74	2x D-type FF	Q-Clock Generation
U12	6N138	opto coupler	MIDI in
U13	74F139	2x 2-to-4 mux	Address Decoding
U14	4364-15	SRAM 8kx8	OSRAM
U15	74F139	2x 2-to-4 mux	Address Decoding
U16	27C256-20	EPROM 32kx8	OSROM high
U17	74LS244	8x buffer/driver	DOC Addressing
U18	4364-15	SRAM 8kx8	DOSRAM
U19	74F139	2x 2-to-4 mux	Address Decoding

*continued on next page*

<i>continued from previous page</i>			
#	Type	Description	Task
U20	27C256-20	EPROM 32kx8	OSROM low
U21	74LS245	8x bidir. buffer/driver	DOC Data
U22	27C512	EPROM 64kx8	WAVE 0
U23	27C512	EPROM 64kx8	WAVE 1
U24	7406	hex inverter	Signal Driver (to Floppy)
U25	LM311	voltage comparator	Tape Input
U26	74LS373	octal latch	Wave Address Demultiplexing
U27	DOC5503	ASIC	Sound Generation, A/D Conversion
U28	74LS377	octal latch	Analog Parts Addressing
U29	AD7524	DAC	CV Generator
U30	TL081	OPAMP	CV amplification
U31	MC34085	OPAMP	Sound Amplifier
U32	TL081	OPAMP	Volume Adjustment
U33	4051	1-to-8 analog mux	Voltage Multiplexer (for ADC)
U34	SSM2300	8x sample & hold	ENV4 CV mux
U35	SSM2300	8x sample & hold	Q CV mux
U36	4051	1-to-8 analog mux	Audio Router
U37	SSM2300	8x sample & hold	PAN CV mux
U38	SSM2300	8x sample & hold	FF VC mux
U39	TL084	4x OPAMP	Audio L/R
U40-U43	CEM3379	analog voice processor	Filter / DCA4 / PAN
U44	CEM3360	dual VCA	final amplifier
U45-U48	CEM3379	analog voice processor	Filter / DCA4 / PAN
U49	74LS05	hex inverters	Floppy Driver
U50	43256C-12	SRAM 32kx8	SEQRAM low
U51	43256C-12	SRAM 32kx8	SEQRAM high
U52	74LS138	3-to-8 mux	SEQRAM select
U53	74LS74	2x D-type FF	Mapper
U54	27C512	EPROM 64kx8	WAVE 2 (not installed)
U55	27C512	EPROM 64kx8	WAVE 3 (not installed)
Y1	OSC8MHz	Quartz Oscillator	System Clock Generator

Table A.1: Parts list

## Appendix B

# Schematics

**Caution:** These schematics do only cover the (vital) digital part of the SQ80. I've spent some time tracing the signals using an ordinary multi-meter and tons of paper... *I do not guarantee the correctness of these self-made schematics so use them entirely at your own risk.*

However, if you encounter any errors feel free to contact me so that I can bugfix the schematics and publish a error-corrected version of this document.





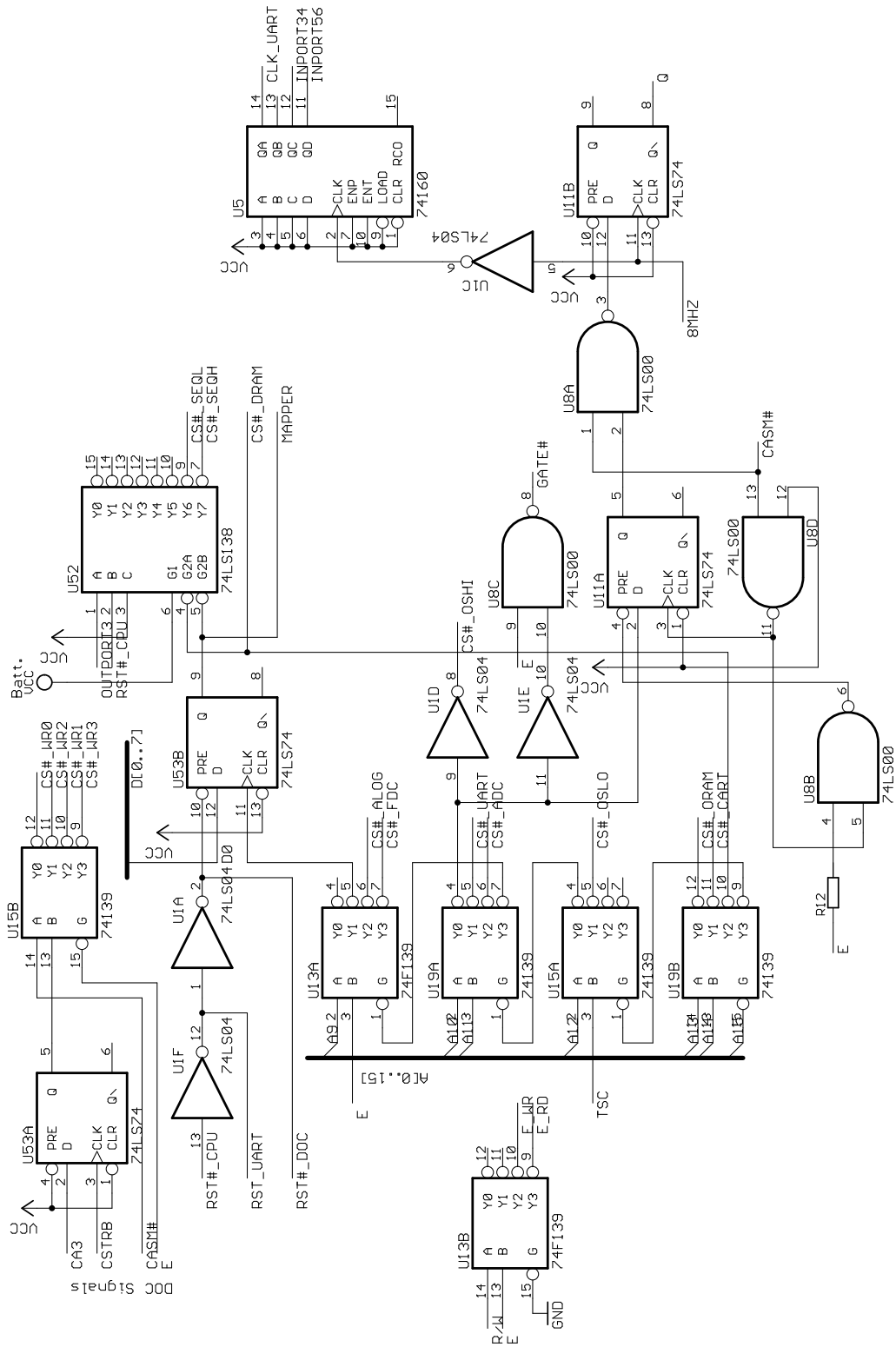


Figure B.1: Memory Decoding & Timing Generator



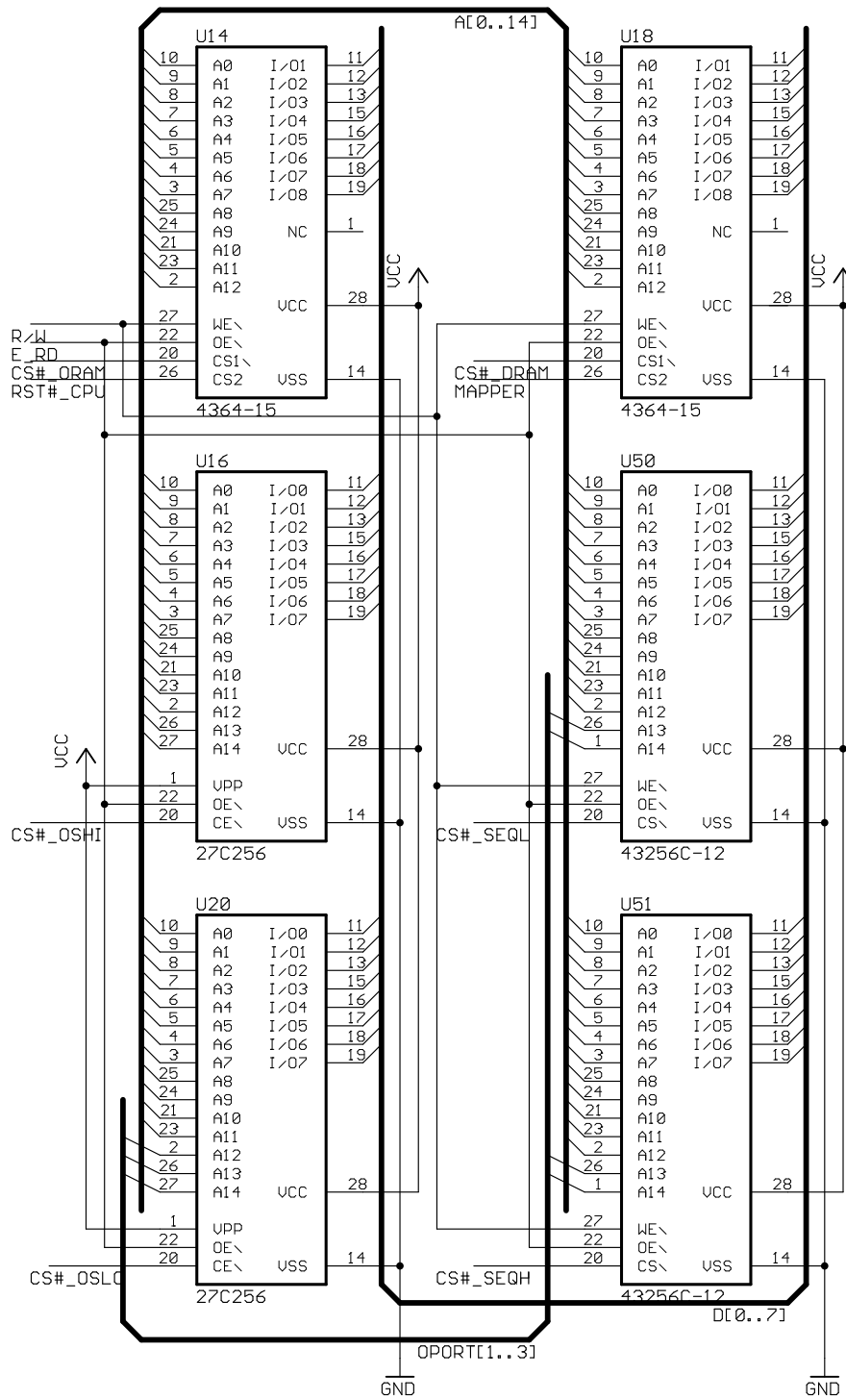


Figure B.2: System Memory



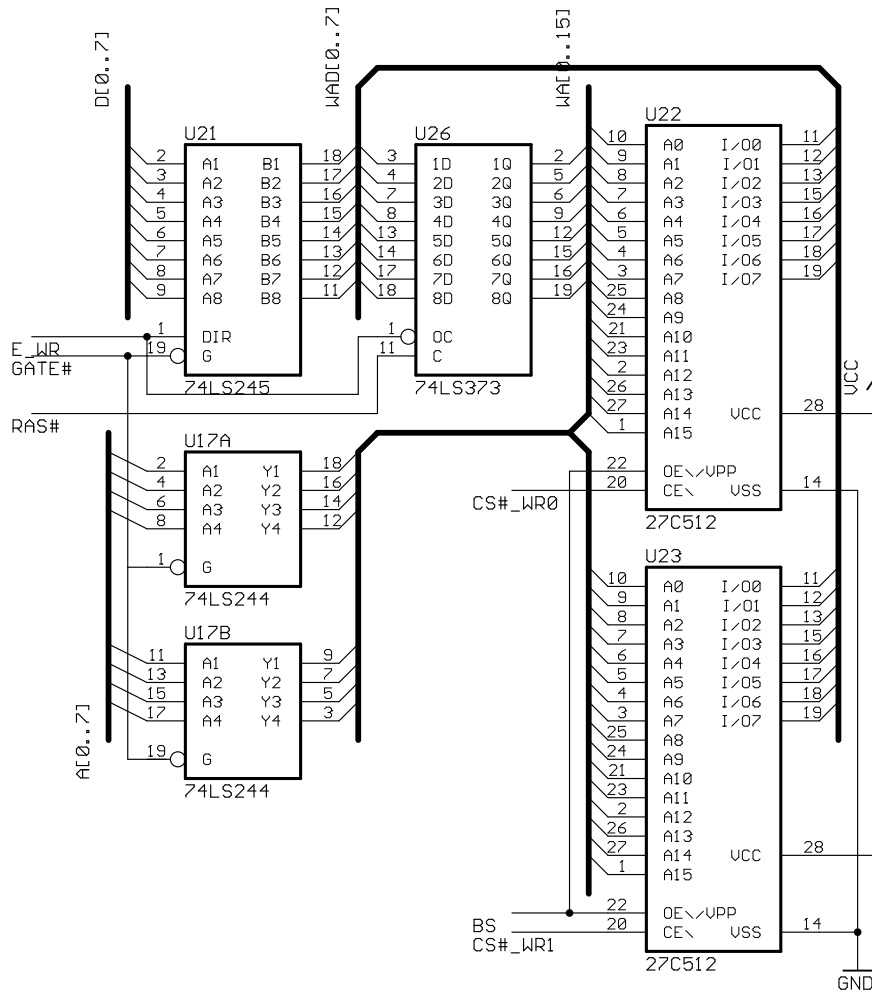


Figure B.3: WaveROM & DOC access



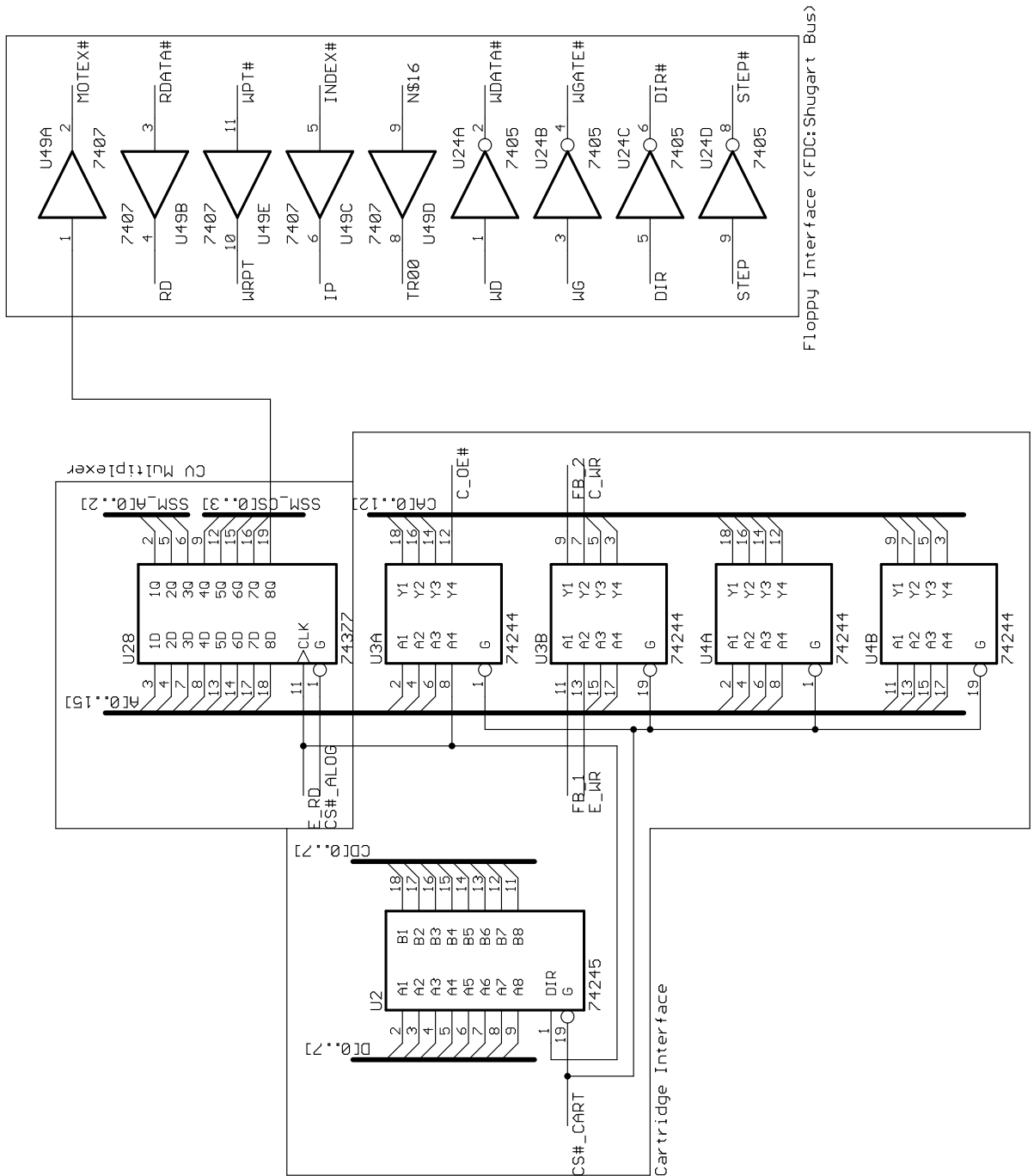


Figure B.4: Peripheral Interfacing





## Appendix C

# Datasheets

For your convenience here are the (some times hard to find) schematics of all vital parts of the SQ80. What I have not included are the common parts such as TTL ICs or OPAMPs which you will find in any usual data book.



## **C.1 WD1770/1772 Floppy Disk Controller / Formatter**

### **C.1.1 Description**

The WD177x is a MOS LSI device which performs the functions of a Floppy Disk Controller/Formatter. It is similar to its predecessor, the WD179x but also contains a digital data separator and write precompensation circuitry. The drive side of the interface needs no additional logic except for buffers/receivers. Designed for single or double density operation. The device contains a programmable Motor On signal.

The WD177x is a low cost version of the FD179x Floppy Disk Controller/Formatter. It is compatible with the 179x, but has a built-in digital data separator and write precompensation circuits. A single read line (RD#, pin 19) is the only input required to recover serial FM or MFM data from the disk drive. The device has been specifically designed for control of floppy disk drives with data rates of 125 Kbits/sec (single density) and 250 Kbits/sec (double density). In addition write precompensation of 125 Nsec from nominal can be enabled at any point through simple software commands. Another programmable feature, Motor On, has been incorporated to enable the spindle motor prior to operating a selected drive.

Two versions of the WD1770 are available. The standard version is compatible with the 179x stepping rates, while the WD1772 offers stepping rates of 2, 3, 5 and 6 msec.

The processor interface consists of an 8-bit bidirectional bus for transfer of status, data and commands. All host communication with the drive occurs through these data lines. They are capable of driving one standard TTL load or three "LS" loads.

### **C.1.2 Architecture**

The Floppy Disk Formatter block diagram is illustrated on page 4. The primary sections include the parallel processor interface the the Floppy Disk Interface.

#### **Data Shift Register**

This 8-bit register assembles serial data from the Read Data input (RD#) during Read operations and transfers serial data to the Write Data output during write operations.

CS#	1	28	INTRQ
R/W	2	27	DRQ
A0	3	26	DDEN#
A1	4	25	WPRT#
D0	5	24	IP#
D1	6	23	TR00#
D2	7	22	WD
D3	8	21	WG
D4	9	20	MO
D5	10	19	RD#
D6	11	18	CLK
D7	12	17	DIR
MR#	13	16	STEP
GND	14	15	Vcc

Figure C.1: WD177x Pinout

**Data Register**

This 8-bit register is used as a holding register during Disk read and Write operations. In Disk Read operations the assembled data byte is transferred in parallel to the Data register from the Data Shift Register. In Disk Write operations information is transferred in parallel from the the Data Register to the Data Shift Register.

When executing the Seek command the Data Register holds the address of the desired Track position. This register is loaded from the DAL and gated into the DAL under processor control.

**Track Register**

This 8-bit register holds the track number of the current Read/Write head position. It is incremented by one every time the head is stepped in and decremented by one when the head is stepped out (towards track 00). The contents of the register are compared with the recorded track number in the ID field during disk Read, Write and Verify operations. The track register can be loaded from or transferred to the DAL. This register should not be loaded when the device is busy.

**Sector Register (SR)**

This 8-bit register holds address of the desired sector position. The contents of the register are compared with the recorded sector number in the ID field during disk Read or Write operations. The Sector Register contents can be loaded from or transferred to the DAL. This register should not be loaded when the device is busy.

**Command Register (CR)**

This 8-bit register holds the command presently being executed. This register should not be loaded when the device is busy unless the new command is a force interrupt. The command register can be loaded from the DAL, but not read into the DAL.

#### **Status Register (STR)**

This 8-bit register holds device Status information. The meaning of the status bits is a function of the type of command previously executed. This register can be read into the DAL, but not loaded from the DAL.

#### **CRC Logic**

This logic is used to check or to generate the 16-bit Cyclic Redundancy Check (CRC). The polynomial is:  $g(x) = x^{16} + x^{12} + x^5 + 1$ . The CRC includes all information starting with the address mark and up to the CRC characters. The CRC register is preset to ones prior to data being shifted through the circuit.

#### **Arithmetic/Logic Unit (ALU)**

The ALU is a serial comparator, incremter and decremter and is used for register modification and comparisons with the disk recorded ID field.

#### **Timing and Control**

All computer and floppy disk interface controls are generated through this logic. The internal device timing is generated from an external crystal clock. The WD177x has two different modes of operation according to the state of DDEN#. When DDEN# = 0, double density is enabled. When DDEN# = 1, single density is enabled.

#### **AM Detector**

The address mark detector detects ID, data and index address marks during read and write operations.

#### **Data Separator**

A digital data separator consisting of a nng shift register and data window detection logic provides read data and a recovery clock to the AM detector.

### **C.1.3 Processor Interface**

The interface to the processor is accomplished through the eight Data Access Lines (DAL) and associated control signals. The DAL are used to transfer data, Status and Control words out of, or into the WD177x. The DAL are three state buffers that are enabled as output drivers when Chip Select (CS#) and R/W = 1 are active or act as input receivers when CS and R/W = 0 are active.

When transfer of data with the Floppy Disk Controller is required by the host processor, the device address is decoded and **CS#** is made low. The address bits **A1** and **A0**, combined with the signal **R/W** during a Read operation or Write operation are interpreted as selecting the following registers:

<b>A1</b>	<b>A0</b>	<b>Read</b>	<b>Write</b>
0	0	Status Register	Command Register
0	1	Track Register	
1	0	Sector Register	
1	1	Data Register	

Table C.1: WD1770/1772 Registers

During Direct Memory Access (DMA) types of data transfers between the Data Register of the WD177x and the processor, the Data ReQuest (DRQ) output is used in Data Transfer control. This signal also appears as status bit 1 during Read and Write operations.

On Disk Read operations the Data Request is activated (set high) when an assembled serial input byte is transferred in parallel to the Data Register. This bit is cleared when the Data Register is read by the processor. If the Data Register is read after one or more characters are lost, by having new data transferred to the register prior to processor readout, the Lost Data bit is set in the Status Register. The read operation continues until the end of sector is reached.

On Disk Write operations the Data ReQuest is activated when the Data Registers its contents to the Data Shift Register and requires a new data byte. It is reset when the Data Register is loaded with new data by the processor. If new data is not loaded at the time serial byte is required by the Floppy Disk, a byte of zeroes is written on the diskette and the Lost Data bit is set in the Status Register.

At the completion of every command an **INTRQ** is generated. **INTRQ** is reset by either reading the status register or by loading the command register with a new command. In addition **INTRQ** is generated if a Force Interrupt command condition is met.

The WD177x has two modes of operation according to the state of **DDEN#** (pin 26). When **DDEN#** = 1, single density is selected. In either case, the **CLK** input (pin 18) is at 8 MHz.

### C.1.4 General Disk Read Operation

Sector lengths of 128, 256, 512 or 1024 bytes are obtainable in either FM or MFM formats. For FM formats, DDEN# should be placed to logical "1". For MFM formats, DDEN# should be placed to logic "0". Sector lengths are determined at format time by the fourth byte in the ID field.

Sector Length	# of Bytes/Sector
00	128
01	256
02	512
03	1024

Table C.2: Sector Length Settings

The number of sectors per track as far as the WD177x is concerned can be from 1 to 255 sectors. The number of tracks as far as the WD177x is concerned is from 0 to 255 tracks.

### C.1.5 General Disk Write Operation

When writing is to take place on the diskette, the Write Gate (WG) output is activated, allowing current to flow into the Read/Write head. As a precaution to erroneous writing the first data must be loaded into the Data Register in response to a Data ReQuest from the device before the Write Gate signal can be activated.

Writing is inhibited when the WP# (Write Protect) input is a logic low, in which case any Write command is immediately terminated, an interrupt generated and the Write Protection status bit is set.

For Write operations, the WD177x provides Write Gate (WG, pin 21) to enable a write condition and Write Data (WD, pin 22) which consists of a series of active high pulses. These pulses contain both Clock and Data information in FM or MFM. Write Data provides the unique missing clock patterns for recording Address Marks.

The Precomp Enable bit in Write commands allow automatic Write Pre-compensation to take place. The outgoing Write Data Stream is delayed or advanced according to the following table:

Pattern	MFM	FM
x 1 1 0	Early	N/A
<i>continued on next page</i>		

<i>continued from previous page</i>		
Pattern	MFM	FM
x 0 1 1	Late	N/A
0 0 0 1	Early	N/A
1 0 0 0	Late	N/A

Table C.3: Precompensation Scheme

Hereby represent the four pattern bits (from left to right) the previous sent bits, the bit currently being sent and the next bit to send.

Precompensation is typically enabled on the innermost tracks where bit shifts usually occur and bit density is at its maximum.

### C.1.6 Command Description

De WD177x will accept eleven commands. Command words should only be loaded in the Command Register when the Busy status bit is off (Status bit 0). The one exception is the force interrupt command. Whenever a command is being executed, the Busy status bit is set. When a command is completed, an interrupt is generated and the Busy status bit is reset. The status register indicates whether the completed command encountered an error or was fault free. For ease of discussion, commands are divided into four types. Commands and types are summarized in table C.4.

Type	Command	Bits 7-0							
1	Restore	0	0	0	0	h	v	r1	r0
1	Seek	0	0	0	1	h	v	r1	r0
1	Step	0	0	1	u	h	v	r1	r0
1	Step-in	0	1	0	u	h	v	r1	r0
1	Step-out	0	1	1	u	h	v	r1	r0
2	Rd sectr	1	0	0	m	h	E	0	0
2	Wt sectr	1	0	1	m	h	E	P	a0
3	Rd addr	1	1	0	0	h	E	0	0
3	Rd track	1	1	1	0	h	E	0	0
3	Wt track	1	1	1	1	h	E	P	0
4	Forc int	1	1	0	1	i3	i2	i1	i0

Table C.4: Command Summary

Type 1 Commands	
h =	Motor on Flag (bit 3)
h = 0	Enable spin-up Sequence
<i>continued on next page</i>	



<i>continued from previous page</i>			
<b>Type 1 Commands</b>			
h = 1	Disable spin-up Sequence		
v =	Verify Flag (bit 2)		
v = 0	No verify		
v = 1	Verify on destn track		
r1, r0 =	Stepping rate (bits 1, 0)		
	r1	r0	WD1770 WD1772
	0	0	6 ms 2 ms
	0	1	12 ms 3 ms
	1	0	20 ms 5 ms
	1	1	30 ms 6 ms
u =	Update Flag (bit 4)		
u = 0	No update		
u = 1	Update Track Register		

Table C.5: Type 1 Commands Flag Summary

<b>Type 2 Commands</b>	
m =	Multiple Sector Flag (bit 4)
m = 0	Single sector
m = 1	Multiple sector
a0=	Data Address Mark (bit 0)
a0= 0	Write normal Data Mark
a0= 1	Write Deleted Data Mark
E =	30ms Settling Delay (bit 2)
E = 0	No delay
E = 1	Add 30ms Delay
P =	Write Precompensation (bit 1)
P = 0	Enable Write Precomp
P = 1	Disable Write Precomp

Table C.6: Type 2 Commands Flag Summary

<b>Type 4 Commands</b>	
i3-i0	Interrupt condition (bit 3-0)
<i>continued on next page</i>	

<i>continued from previous page</i>	
<b>Type 4 Commands</b>	
i0= 1	Don't care
i1= 1	Don't care
i2= 1	Interrupt on index pulse
i3= 1	Immediate interrupt
13-i0 = 0	Terminate without interrupt

Table C.7: Type 4 Commands Flag Summary

### C.1.7 Type 1 Commands

The type 1 commands include the Restore, Seek, Step, Step-in and Step-out commands. Each of the Type 1 commands contains a rate field (r0, r1), which determines the stepping motor rate.

A  $4\mu\text{s}$  (MFM) or  $8\mu\text{s}$  (FM) pulse is provided as an output to the drive. For every step pulse issued, the drive moves one track location in a direction determined by the direction output. The chip will step the drive in the same direction it last stepped, unless the command changes the direction.

The direction signal is active high when stepping in and low when stepping out. The Direction signal is valid 24 $\mu\text{s}$  before the first stepping pulse is generated.

After the last directional step an additional 30 milliseconds of head settling time takes if the verify flag is set in type 1 commands. There is also a 30ms head settling time if the E flag is set in any Type 2 or 3 command.

When a Seek, Step or Restore command is executed, an optional verification of Read/Write head position can be performed by setting bit 2 (V=1) in the command word to a logic "1". The verification operation begins at the end of the 30ms settling time after the head is loaded against the media. The track number from the first encountered ID field is compared against the contents of the Track Register. If the track numbers compare and the ID field CRC is correct, the verify operations is complete and an INTRQ is generated with no errors. If there is a match but not a valid CRC, the CRC error status bit is set (Status bit 3) and the next encountered ID field is read from the disk for the verification operation.

The WD177x must find an ID field with correct track number and correct CRC within 5 revolutions of the media, otherwise the seek error is set and an INTRQ is generated. If V=0 no verification is performed.

All commands except the Force Interrupt command may be programmed via the h Flag to delay for spindle motor startup time. If the h Flag is set and the Motor On line (M0, pin 20) is low when a command is received, the WD177x will force Motor On to a logic "1" and wait 6 revolutions before executing the command. At 300rpm this guarantees a one second spindle startup time. If after finishing the command, the device remains idle for 10 revolutions, the Motor On line will go back to a logic "0". If a command is issued while Motor On is high, the command will execute immediately, defeating the 6 revolutions start up. This feature allows consecutive Read or Write commands without waiting for motor start up each time; the WD177x assumes the spindle motor is up to speed.

### **Restore (Seek Track 0)**

Upon receipt of this command, the Track 00 (TR00#) input is sampled. If TR00# is active low indicating the Read/Write head is positioned over track 0, the Track register is loaded with zeroes and an interrupt is generated. If TR00# is not active low, stepping pulses (pin 16) at a rate specified by the r1, r0 field are issued until the TR00# input is activated. At this time, the Track Register is loaded with zeroes and an interrupt is generated. If the TR00# input does not go active low after 255 stepping pulses, the WD177x terminates operation, interrupts and sets the Seek error status bit, providing the V flag is set. A verification also takes place if the V flag is set. The h bit allows the Motor On option at the start of command.

### **Seek**

This command assumes that the track register contains the track number of the current position of the Read/Write head and the Data Register contains the desired track number. The WD177x will update the Track Register and issue stepping pulses in the appropriate direction until the contents of the Track Register are equal to the contents of the Data Register (the desired Track location). A verification operation takes place if the V flag is on. The h bit allows the Motor On option at the start of the command. An interrupt is generated at the completion of the command. Note: When using multiple drives, the track register must be updated for the drive selected before seeks are issued.

### **Step**

Upon receipt of this command, the WD177x issues one stepping pulse to the disk drive. The stepping motor direction is the same as in the previous step command. After a delay determined by the r1, r0 field, a verification takes place if the V flag is on. If the U flag is on, the Track Register is updated. The h bit allows the Motor On option at

the start of the command. An interrupt is generated at the completion of the command.

#### **Step-In**

Upon receipt of this command, the WD177x issues one stepping pulse in the direction towards track 76. If the U flag is on, the Track Register is incremented by one. After a delay determined by the r1, r0 field, a verification takes place if the V flag is on. The h bit allows the Motor On option at the start of the command. An interrupt is generated at the completion of the command

#### **Step-Out**

Upon receipt of this command, the WD177x issues one stepping pulse in the direction towards track 0. If the U flag is on, the Track Register is decremented by one. After delay determined by the r1, r0 field, a verification takes place if the V flag is on. The h bit allows the Motor On option at the start of the command.

### **C.1.8 Type 2 Commands**

The type 2 commands are the read sector and write sector commands. Prior to loading the type 2 command into the command register, the computer must load the sector register with the desired sector number. Upon receipt of the type 2 command, the busy status bit is set. If the E flag is 1, the command will execute after 30ms delay.

When an ID field is located on the disk, the WD177x compares the track number on the ID field with the Track register. If there is not a match, the next encountered ID field is read and a comparison is again made. If there was a match, the Sector Number of the ID field is compared with the Sector Register. If there is not a Sector match, the next encountered ID field is read off the disk and comparisons again made. If the ID field CRC is correct, the data field is then located and will be either written into, or read from depending upon the command. The WD177x must find an ID field with a track number, sector number and CRC within four revolutions of the disk, otherwise, the Record not found status bit is set (status bit 4) and the command is terminated with an interrupt (INTRQ).

Each of the type 2 commands contains an (m) flag which determines if multiple records (sectors) are to be read or written, depending upon the command. If m=0, a single record is read or written and an interrupt is generated at the completion of the command. If m=1, multiple records are read or written with the sector register internally updated so that an address verification can occur on the next record. The WD177x will continue to read or write multiple records and update the sector register in numerical

ascending sequence until the sector number exceeds the number of sectors on the track or until the Force Interrupt command is loaded in the command register, which terminates the command and generates an interrupt.

**Example:** If the WD177x is instructed to read sector 17 and there are only 16 sectors on the track, the sector register exceeds the number available. The WD177x will search for 5 disk revolutions, interrupt out, reset busy and set the record not found status bit.

### Read Sector

Upon receipt of the Read Sector command, the busy status bit is set, and when an ID field is encountered that has the correct track number, correct sector number and correct CRC, the data field is presented to the computer. The data address mark of the data field must be found within 30 bytes in single density and 43 bytes in double density of the last ID field CRC byte; if not, the ID field is searched for and verified again followed by the Data Address Mark search. If after 5 revolutions the DAM cannot be found, the record not found bit is set and the operation terminated. When the first character or byte of the data field has been shifted through the DSR, it is transferred to the DR, and DRQ is generated. When the next byte is accumulated in the DSR, it is transferred to the DR and another DRQ is generated. If the computer has not read the previous contents of the DR before a new character is transferred that character will be lost and the lost data status bit is set. This sequence continues until the complete data field has been inputted to the computer. If there is a CRC error at the end of the data field, the CRC error status bit is set and the command is terminated (even if it is a multiple record command).

At the end of the Read operation, the type of Data Address Mark encountered in the data field is recorded in the status register (bit 5) as shown:

1 represents a deleted data mark whereas

0 represents a data mark

### Write Sector

Upon receipt of the Write Sector command, the Busy status bit is set. When an ID field is encountered that has the correct track number, correct sector number and correct CRC, a DRQ is generated. The WD177x counts off 11 bytes in single density and 22 bytes in double density from the CRC field and the Write Gate (WG) output is made active if the DRQ is serviced (i.e. the DR has been loaded by the computer). If DRQ has not been serviced, the command is terminated and the lost data status bit is set. If the DRQ has been serviced, the

WG is made active and six bytes of zeroes in single density and 12 bytes in double density are then written to the disk. At this time, the Data Address Mark is then written on the disk as determined by the a0 field of the command as shown below:

**1** represents a deleted data mark whereas

**0** represents a data mark

The WD177x then writes the data field and generates DRQ's to the computer. If the DRQ is not serviced in time for continuous writing the lost data status bit is set and a byte of zeroes is written on the disk. The command is not terminated. After the last data byte has been written on the disk, the two byte CRC is computed internally and written on the disk followed by one byte of logic ones in FM or MFM. The WG output is then activated. **INTRQ** will set 24uSec (MFM) after the last CRC byte is written. For partial sector writing, the proper method is to write data and fill the balance with zeroes.

### C.1.9 Type 3 Commands

#### Read Address

Upon receipt of the Read Address command, the Busy status bit is set. The next encountered ID field is then read in from the disk, and six data bytes of the ID field are assembled and transferred to the DR, and a DRQ is generated for each byte. The six bytes of the ID field are shown below.

Track Address	Side Number	Sector Address	Sector Length	CRC 1	CRC 2
1	2	3	4	5	6

Table C.8: ID Field

Although the CRC characters are transferred to the computer, the WD177x checks for validity and the CRC error status bit if there is a CRC error. The Track Address of the ID field is written into the sector register so that a comparison can be made by the user. At the end of the operation an interrupt is generated and the Busy status bit is reset.

#### Read Track

Upon receipt of the Read Track command, the head is loaded and the Busy status bit is set. Reading starts with the leading edge of the first encountered index pulse and continues until the next index pulse. All gap, Header and data bytes are assembled and transferred to the data

register and DRQ's are generated for each byte. The accumulation of bytes is synchronized to each address mark encountered. An interrupt is generated at the completion of the command. This command has several characteristics which makes it suitable for diagnostic purposes. They are: no CRC checking is performed; gap information is included in the datastream; and the address mark detector is on for the duration of the command. Because the AM detector is always on, write splices or noise may cause the chip to look for an DM.

The ID am, ID field, ID CRC bytes, DAM, data, and data CRC bytes for each sector will be correct. The Gap Bytes may be read incorrectly during write-splice time because of synchronization.

### Write Track / Formatting the Disk

Formatting the disk is a relatively simple task when operating programmed I/O or when operating under DMA with a large amount of memory. Data and gap information must be provided at the computer interface. Formatting the disk is accomplished by positioning the R/W head over the desired track and issuing the Write Track command.

Upon receipt of the Write Track command, the Busy status bit is set. Writing starts with the leading edge of the first encountered index pulse and continues until the next index pulse, at which time the interrupt is activated. The Data ReQuest is activated immediately upon receiving the command, but writing will not start until after the first byte has been loaded into the Data Register. If the DR has not been loaded within 3 byte times, the operation is terminated making the device not busy, the Lost Data Status bit is set, and the interrupt is activated. If a byte is not present in the DR when needed, a byte of zeroes is substituted.

This sequence continues from one index mark to the next index mark. Normally, whatever data pattern appears in the data register is written on the disk with a normal clock pattern. However, if the WD177x detects a data pattern of F5 through FE in the data register, this is interpreted as data address marks with missing clocks or CRC generation.

<b>Data Pattern in DR (hex)</b>	<b>in FM (DDEN#=1)</b>	<b>in MFM (DDEN#=0)</b>
00 thru F4 F5	Wt 00 thru F4 with CLK = FF Not Allowed	Wt 00 thru F4 in MFM Wt A1 in MFM, Preset CRC
<i>continued on next page</i>		

<i>continued from previous page</i>		
<b>Data Pattern in DR (hex)</b>	<b>in FM (DDEN#=1)</b>	<b>in MFM (DDEN#=0)</b>
F6	Not Allowed	Wt C2 in MFM
F7	Generate 2 CRC bytes	Generate 2 CRC bytes
F8 thru FB	Wt F8-FB, Clk=C7, Preset CRC	Wt F8 thru Fb, in MFM
FC	Wt FC with Clk=D7	Wt FC in MFM
FD	Wt FD with Clk=FF	Wt FD in MFM
FE	Wt FE, Clk=C7, Preset CRC	Wt FE in MFM
FF	Wt FF with Clk=FF	Wt FF in MFM

Table C.9: ID Field

The CRC generator is initialized when any data byte from F8 to FE is about to be transferred from the DR to the DSR in FM or by receipt of F5 in MFM. An F7 pattern will generate two CRC characters in FM or MFM. As a consequence, the patterns F5 through FE must not appear in the gaps, data fields or ID fields. Also, CRC's must be generated by an F7 pattern.

Disks may be formatted in IBM 3740 or system 34 formats with sector lengths of 128, 256, 512 or 1024 bytes.

### C.1.10 Type 4 Commands

The forced interrupt command is generally used to terminate a multiple sector read or write command or to insure Type 1 status in the status register. This command can be loaded into the command register at any time. If there is a current command under execution (Busy status bit set) the command will be terminated and the busy status bit reset.

The lower four bits of the command determine the conditional interrupt as follows:

- i0** Don't care
- i1** Don't care
- i2** Every index puls
- i3** Immediate interrupt

The conditional interrupt is enabled when bit positions of the command (i3-i0) are set to a "1". Then, when the condition for interrupt is met, the INTRQ line will go high signifying that the condition specified has occurred. If i3-i0 are all set to zero (\$d0), no interrupt will occur, but any command



presently under execution will be immediately terminated. When using the immediate interrupt condition ( $i3=1$ ) an interrupt will be immediately generated and the current command terminated. Reading the status or writing to the command register will not automatically clear the interrupt.  $\$d0$  is the only command that will enable the immediate interrupt ( $\$d6$ ) to clear on a subsequent load command register or read status register operation. Follow a  $\$d6$  with  $\$d0$  command.

Wait  $16\mu s$  (double density) or  $32\mu s$  (single density) before issuing a new command after issuing the forced interrupt. Loading a new command sooner than this will nullify the forced interrupt.

Forced interrupt stops any command at the end of an internal micro-instruction and generates **INTRQ** when the specified condition is met. Forced interrupt will wait until ALU operations in progress are complete (CRC calculations, compares, etc.).

### C.1.11 Status Register

Upon receipt of any command, except the Force interrupt command the Busy status bit is set and the rest of the status bits are updated or cleared for the new command. If the Force Interrupt Command is received when there is a current command under execution, the Busy status bit is reset and the rest of the status bits are unchanged. If the Forced interrupt command is received when there is not a current command under execution, the Busy status bit is reset and the rest of the status bits are updated or cleared. In this case, Status reflects the type 1 commands.

The user has the option of reading the status register through program control or using the DRQ with DMA or interrupt methods. When the Data register is read the DRQ bit in the status register and the DRQ line are automatically reset. A write to the Data Register also causes both DRQ's to reset.

The busy bit in the status may be monitored with a user program to determine when a command is complete, in lieu of using the **INTRQ** line. When using the **INTRQ**, a busy status check is not recommended because a read of the status register to determine the condition of busy will reset the **INTRQ** line.

The format of the Status register is shown in the following table:

Bit	Name	Meaning
7	MOTOR ON	This bit reflects the status of the Motor On output.
6	WRITE PROTECT	On read record: not used. On read track: not used. On any write: it indicates a Write Protect. This bit is reset when updated.
5	SPIN-UP	When set, this bit indicates that the Motor Spin-Up sequence has completed (6 revolutions) on type 1 commands.
	RECORD-TYPE	On Type 2 & 3 commands, this bit indicates record Type. 0 = Data Mark, 1 = Deleted Data Mark.
4	RECORD NOT FOUND	When set, it indicates that the desired track, sector, or side were not found. This bit is reset when updated.
3	CRC ERROR	If bit 4 is set, an error is found in one or more ID fields; otherwise it indicates error in data field. This bit is reset when updated.
2	LOST DATA	When set, it indicates the computer did not respond to DRQ in one byte time. This bit is reset to zero when update.
	TRACK00	On type 1 commands, this bit reflects the status of the TR00# pin.
1	DATA REQUEST	This bit is a copy of the DRQ output. When set, it indicates the DR is full on a Read Operation or the DR is empty on a write operation. This bit is reset to zero when updated.
	INDEX	On type 1 commands, this bit indicates the status of the index pin.
0	BUSY	When set, command is under execution. When reset, no command is under execution.

Table C.10: Status Register

### C.1.12 Recommended Layout for 128-Byte Sectors

Shown below is the recommended single-density format with 128 bytes/sector. In order to format a diskette, the user must issue the Write Track command and load the data register with the following values. For every byte to be written there is one Data Request.

# of bytes	Value of Byte(s)	Meaning
40	\$ff (or \$00)	GAP1
6	\$00	GAP3
1	\$fe	ID field Address Mark
1	tn	Track number
1	hn	Side number (0 or 1)
1	sn	Sector number (0 thru 25)
1	\$00	Sector length
1	\$f7	CRC
11	\$ff (or \$00)	GAP2
6	\$00	GAP2
1	\$fb	Data Address Mark
128	data	(IBM uses \$e5)
1	\$f7	CRC
10	\$ff (or \$00)	GAP4a
369	\$ff (or \$00)	GAP4b

Table C.11: ID Field for 128-Byte Sectors

The enclosed area is written 16 times. GAP4 has to be written until WD1772 interrupts out, 369 bytes as an approximate value.

### C.1.13 Recommended Layout for 256-Byte Sectors

Shown below is the recommended dual-density format with 256 bytes/sector. In order to format a diskette the user must issue the Write Track command and load the data register with the following values. For every byte to be written there is one data request.

# of bytes	Value of Byte(s)	Meaning
40	\$4e	GAP1
12	\$00	GAP3
3	\$f5	writes A1, GAP3
1	\$fe	ID field Address Mark
1	tn	Track number
1	hn	Side number (0 or 1)
1	sn	Sector number (0 thru 25)
1	\$01	Sector length
1	\$f7	CRC
22	\$4e	GAP2
12	\$ff (or \$00)	GAP2
3	\$f5	writes \$A1, GAP2

*continued on next page*

<i>continued from previous page</i>		
# of bytes	Value of Byte(s)	Meaning
1	\$fb	Data Address Mark
256	data	(IBM uses \$e5)
1	\$f7	CRC
24	\$4e	GAP4a
668	\$4e	GAP4b

Table C.12: ID Field for 256-Byte Sectors

The enclosed area is written 16 times. GAP4 has to be written until WD1772 interrupts out, 668 bytes as an approximate value.

### C.1.14 Generic (non-standard) formats

Variations in the recommended formats are possible to a limited extent if the following requirements are met.

1. Sector length must be 128, 256, 512 or 1024 bytes.
2. Gap 2 cannot be varied from the recommended format.
3. 3 bytes of \$a1 must be used in MFM.

In addition, the index Address Mark is not required for operation by the WD177x. Gap 1, 3 and 4 lengths can be as short as 2 bytes for WD177x operation, however PLL lock up time, motor speed variation, write splice area, etc. will add more bytes to each gap to achieve proper operation. It is recommended that the recommended format be used for highest system reliability.

# of bytes	Value of Byte	Comments
60	\$4e	Gap 1 and Gap 3 Start and end of index pulse.
12	\$00	Gap 3 Start of bytes repeated for each sector
3	\$a1	Gap 3 Start of ID field (see C.1.9)
1	\$fe	ID address mark (IDAM)
1	track #	Track number
1	side #	Head number (0 or 1)
1	sector #	Sector number (0 to 25)
1	length code	Sector length (see C.2)

*continued on next page*

<i>continued from previous page</i>		
# of bytes	Value of Byte	Comments
2	CRC	END of ID field (see C.1.9)
22	\$4e	Gap 2
12	\$00	Gap 2
		During Write Sector commands the drive starts writing at the beginning of this.
3	\$a1	Gap 2
		Start of data field (see C.1.9)
1	\$fb	Data Address Mark (DAM)
sector size	data	Values \$f5 to \$f7 are invalid. (see C.1.9)
2	CRC	End of data field. (see C.1.9)
24	\$4e	Gap 4
		During Write Sector Commands the drive stops writing shortly after the beginning of this.
sector size*2.5	\$4e	Continue writing until the 177x generates an interrupt. The listed byte count is approximate.

Table C.13: Gap settings for non-standard formats

[Command Flow Diagrams left out]



## C.2 SSM2300 Octal Sample&Hold

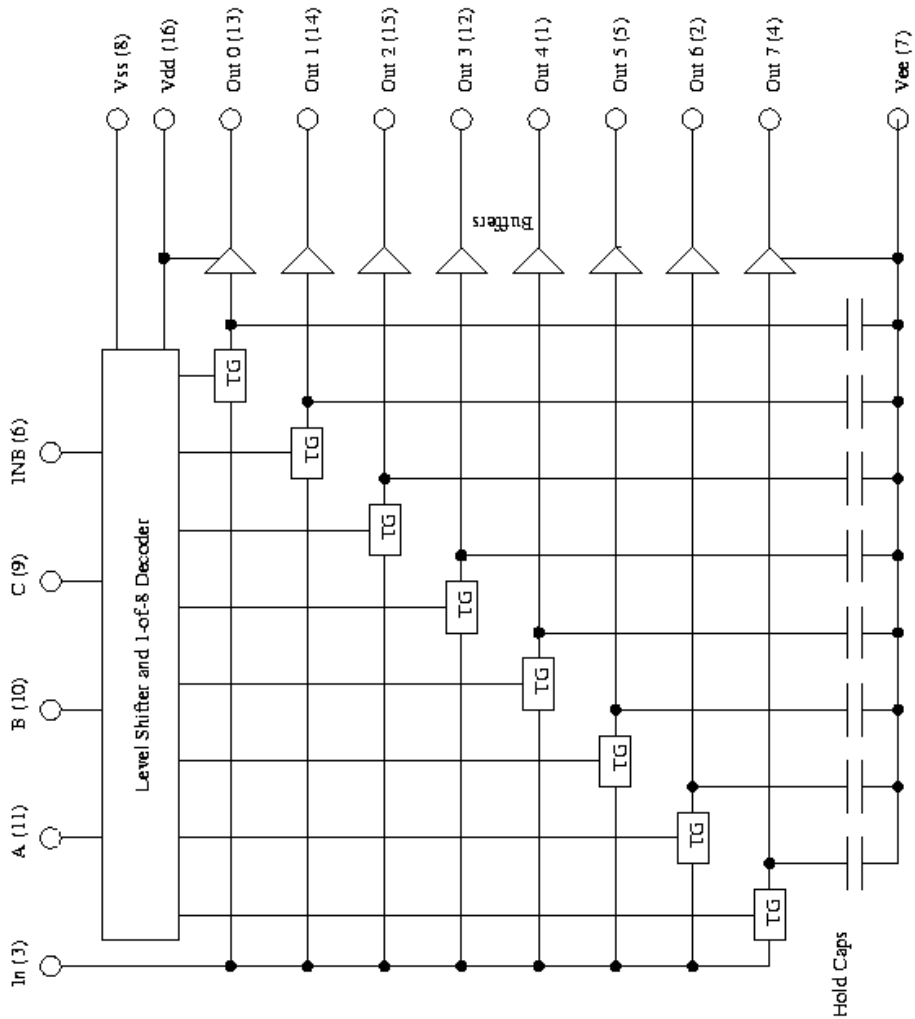


Figure C.2: SSM2300 Octal Sample&Hold





### **C.3 MC/SN 2681 DUART**



**Dual asynchronous receiver/transmitter (DUART)****SCN2681****DESCRIPTION**

The Philips Semiconductors SCN2681 Dual Universal Asynchronous Receiver/Transmitter (DUART) is a single-chip MOS-LSI communications device that provides two independent full-duplex asynchronous receiver/transmitter channels in a single package. It interfaces directly with microprocessors and may be used in a polled or interrupt driven system.

The operating mode and data format of each channel can be programmed independently. Additionally, each receiver and transmitter can select its operating speed as one of eighteen fixed baud rates, a 16X clock derived from a programmable counter/timer, or an external 1X or 16X clock. The baud rate generator and counter/timer can operate directly from a crystal or from external clock inputs. The ability to independently program the operating speed of the receiver and transmitter make the DUART particularly attractive for dual-speed channel applications such as clustered terminal systems.

Each receiver is quadruply buffered to minimize the potential of receiver over-run or to reduce interrupt overhead in interrupt driven systems. In addition, a flow control capability is provided to disable a remote DUART transmitter when the buffer of the receiving device is full.

Also provided on the SCN2681 are a multipurpose 7-bit input port and a multipurpose 8-bit output port. These can be used as general purpose I/O ports or can be assigned specific functions (such as clock inputs or status/interrupt outputs) under program control.

The SCN2681 is available in three package versions: 40-pin and 28-pin, both 0.6" wide DIPs; a compact 24-pin 0.4" wide DIP; and a 44-pin PLCC.

**FEATURES**

- Dual full-duplex asynchronous receiver/transmitter
- Quadruple buffered receiver data registers
- Programmable data format
  - 5 to 8 data bits plus parity
  - Odd, even, no parity or force parity
  - 1, 1.5 or 2 stop bits programmable in 1/16-bit increments
- Programmable baud rate for each receiver and transmitter selectable from:
  - 22 fixed rates: 50 to 115.2k baud

- 16-bit programmable Counter/Timer
  - Non-standard rates to 115.2Kb
  - One user-defined rate derived from programmable timer/counter
  - External 1X or 16X clock
- Parity, framing, and overrun error detection
- False start bit detection
- Line break detection and generation
- Programmable channel mode
  - Normal (full-duplex)
  - Automatic echo
  - Local loopback
  - Remote loopback
- Multi-function programmable 16-bit counter/timer
- Multi-function 7-bit input port
  - Can serve as clock or control inputs
  - Change of state detection on four inputs
  - 100kΩ typical pull-up resistor
- Multi-function 8-bit output port
  - Individual bit set/reset capability
  - Outputs can be programmed to be status/interrupt signals
- Versatile interrupt system
  - Single interrupt output with eight maskable interrupting conditions
  - Output port can be configured to provide a total of up to six separate wire-ORable interrupt outputs
- Maximum data transfer: 1X – 1MB/sec, 16X – 125KB/sec
- Automatic wake-up mode for multidrop applications
- Start-end break interrupt/status
- Detects break which originates in the middle of a character
- On-chip crystal oscillator
- Single +5V power supply
- Commercial and industrial temperature ranges available
- DIP and PLCC packages

**ORDERING INFORMATION**

DESCRIPTION	ORDER CODE					
	Commercial			Industrial		
	V <sub>CC</sub> = +5V ±5%, T <sub>A</sub> = 0°C to +70°C					
	Ceramic DIP	Plastic DIP	Plastic LCC	Ceramic DIP	Plastic DIP	Plastic LCC
24-Pin <sup>1</sup>	Not available	SCN2681AC1N24	Not available	Not available	SCN2681AE1N24	Not available
28-Pin <sup>2</sup>	SCN2681AC1F28	SCN2681AC1N28	Not available	SCN2681AE1F28	SCN2681AE1N28	Not available
40-Pin <sup>2</sup>	Not available	SCN2681AC1N40	Not available	SCN2681AE1F40	SCN2681AE1N40	Not available
44-Pin	Not available	Not available	SCN2681AC1A44	Not available	Not available	SCN2681AE1A44

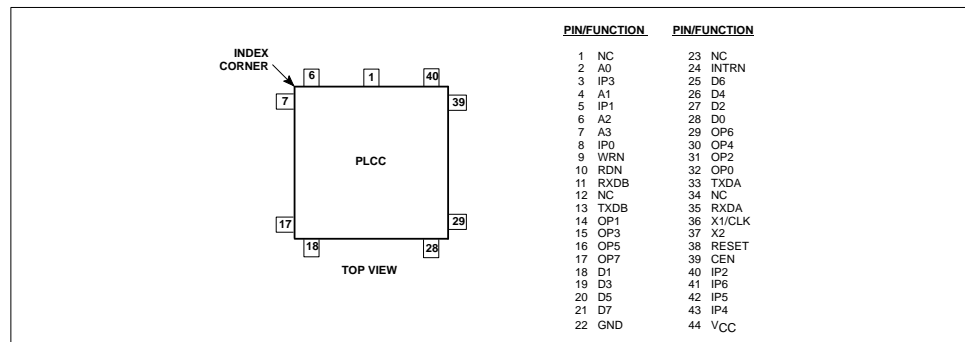
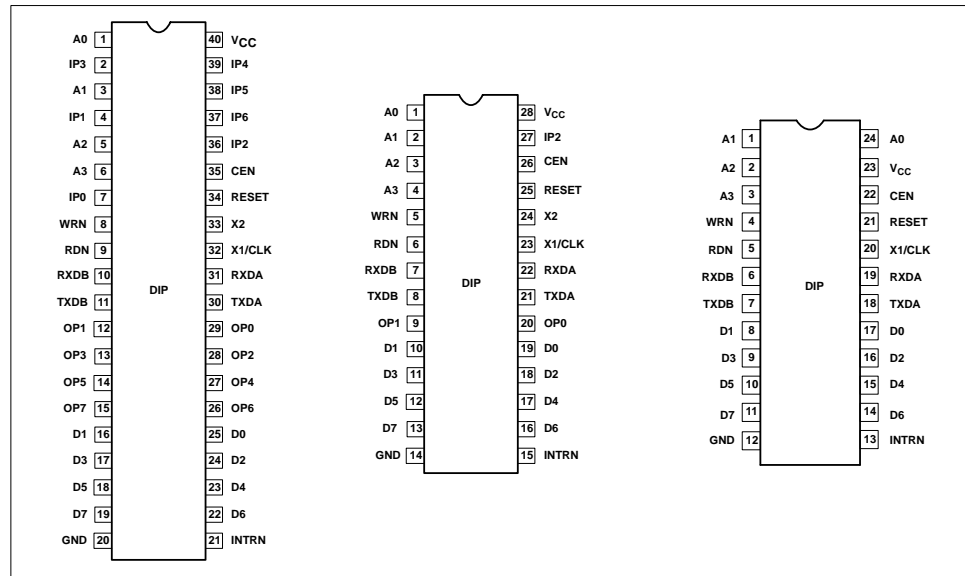
**NOTES:**

1. 400mil-wide Dual In-Line Package
2. 600mil-wide Dual In-Line Package

Dual asynchronous receiver/transmitter (DUART)

SCN2681

PIN CONFIGURATIONS



SD00084

## Dual asynchronous receiver/transmitter (DUART)

SCN2681

## PIN DESCRIPTION

SYMBOL	APPLICABLE			TYPE	NAME AND FUNCTION
	40/44	28	24		
D0-D7	X	X	X	I/O	<b>Data Bus:</b> Bidirectional 3-State data bus used to transfer commands, data and status between the DUART and the CPU. D0 is the least significant bit.
CEN	X	X	X	I	<b>Chip Enable:</b> Active-Low input signal. When Low, data transfers between the CPU and the DUART are enabled on D0-D7 as controlled by the WRN, RDN and A0-A3 inputs. When High, places the D0-D7 lines in the 3-State condition.
WRN	X	X	X	I	<b>Write Strobe:</b> When Low and CEN is also Low, the contents of the data bus is loaded into the addressed register. The transfer occurs on the rising edge of the signal.
RDN	X	X	X	I	<b>Read Strobe:</b> When Low and CEN is also Low, causes the contents of the addressed register to be presented on the data bus. The read cycle begins on the falling edge of RDN.
A0-A3	X	X	X	I	<b>Address Inputs:</b> Select the DUART internal registers and ports for read/write operations.
RESET	X	X	X	I	<b>Reset:</b> A High level clears internal registers (SRA, SRB, IMR, ISR, OPR, OPCR), puts OP0-OP7 in the High state, stops the counter/timer, and puts Channels A and B in the inactive state, with the TxDA and TxDB outputs in the mark (High) state. Clears Test modes, sets MR pointer to MR1.
INTRN	X	X	X	O	<b>Interrupt Request:</b> Active-Low, open-drain, output which signals the CPU that one or more of the eight maskable interrupting conditions are true.
X1/CLK	X	X	X	I	<b>Crystal 1:</b> Crystal or external clock input. A crystal or clock of the specified limits must be supplied at all times. When a crystal is used, a capacitor must be connected from this pin to ground (see Figure 5).
X2	X	X		I	<b>Crystal 2:</b> Connection for other side of the crystal. When a crystal is used, a capacitor must be connected from this pin to ground (see Figure 5).
RxDA	X	X	X	I	<b>Channel A Receiver Serial Data Input:</b> The least significant bit is received first. "Mark" is High, "space" is Low.
RxDB	X	X	X	I	<b>Channel B Receive Serial Data Input:</b> The least significant bit is received first. "Mark" is High, "space" is Low.
TxDA	X	X	X	O	<b>Channel A Transmitter Serial Data Output:</b> The least significant bit is transmitted first. This output is held in the "mark" condition when the transmitter is disabled, idle or when operating in local loopback mode. "Mark" is High, "space" is Low.
TxDB	X	X	X	O	<b>Channel B Transmitter Serial Data Output:</b> The least significant bit is transmitted first. This output is held in the "mark" condition when the transmitter is disabled, idle or when operating in local loopback mode. "Mark" is High, "space" is Low.
OP0	X	X		O	<b>Output 0:</b> General purpose output or Channel A request to send (RTSAN, active-Low). Can be deactivated automatically on receive or transmit.
OP1	X	X		O	<b>Output 1:</b> General purpose output or Channel B request to send (RTSBN, active-Low). Can be deactivated automatically on receive or transmit.
OP2	X			O	<b>Output 2:</b> General purpose output or Channel A transmitter 1X or 16X clock output, or Channel A receiver 1X clock output.
OP3	X			O	<b>Output 3:</b> General purpose output or open-drain, active-Low counter/timer output or Channel B transmitter 1X clock output, or Channel B receiver 1X clock output.
OP4	X			O	<b>Output 4:</b> General purpose output or Channel A open-drain, active-Low, RxRDYA/FFULLA output.
OP5	X			O	<b>Output 5:</b> General purpose output or Channel B open-drain, active-Low, RxRDYB/FFULLB output.
OP6	X			O	<b>Output 6:</b> General purpose output or Channel A open-drain, active-Low, TxRDYA output.
OP7	X			O	<b>Output 7:</b> General purpose output or Channel B open-drain, active-Low, TxRDYB output.
IP0	X			I	<b>Input 0:</b> General purpose input or Channel A clear to send active-Low input (CTSAN).
IP1	X			I	<b>Input 1:</b> General purpose input or Channel B clear to send active-Low input (CTSBN).
IP2	X	X		I	<b>Input 2:</b> General purpose input or counter/timer external clock input.
IP3	X			I	<b>Input 3:</b> General purpose input or Channel A transmitter external clock input (TxCA). When the external clock is used by the transmitter, the transmitted data is clocked on the falling edge of the clock.
IP4	X			I	<b>Input 4:</b> General purpose input or Channel A receiver external clock input (RxCA). When the external clock is used by the receiver, the received data is sampled on the rising edge of the clock.

## Dual asynchronous receiver/transmitter (DUART)

SCN2681

## PIN DESCRIPTION (Continued)

SYMBOL	APPLICABLE			TYPE	NAME AND FUNCTION
	40/44	28	24		
IP5	X			I	<b>Input 5:</b> General purpose input or Channel B transmitter external clock input (TxCB). When the external clock is used by the transmitter, the transmitted data is clocked on the falling edge of the clock.
IP6	X			I	<b>Input 6:</b> General purpose input or Channel B receiver external clock input (RxCB). When the external clock is used by the receiver, the received data is sampled on the rising edge of the clock.
V <sub>CC</sub>	X	X		I	<b>Power Supply:</b> +5V supply input.
GND	X	X		I	<b>Ground</b>

ABSOLUTE MAXIMUM RATINGS<sup>1</sup>

SYMBOL	PARAMETER	RATING	UNIT
T <sub>A</sub>	Operating ambient temperature range <sup>2</sup>	See Note 4	°C
T <sub>STG</sub>	Storage temperature range	-65 to +150	°C
	All voltages with respect to ground <sup>3</sup>	-0.5 to +6.0	V

## NOTES:

- Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operation section of this specification is not implied.
- For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
- This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
- Parameters are valid over specified temperature range. See Ordering information table for applicable operating temperature range and V<sub>CC</sub> supply range.

DC ELECTRICAL CHARACTERISTICS<sup>1, 2, 3</sup>

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V <sub>IL</sub>	Input low voltage		2		0.8	V
V <sub>IH</sub>	Input high voltage (except X1/CLK) <sup>5</sup>		2.5			V
V <sub>IH</sub>	Input high voltage (except X1/CLK) <sup>4</sup>		4			V
V <sub>OL</sub>	Output low voltage	I <sub>OL</sub> = 2.4mA			0.4	V
V <sub>OH</sub>	Output high voltage (except o.d. outputs) <sup>5</sup>	I <sub>OH</sub> = -400µA	2.4			V
V <sub>OH</sub>	Output high voltage (except o.d. outputs) <sup>4</sup>	I <sub>OH</sub> = -400µA	2.9			V
I <sub>IL</sub>	Input leakage current	V <sub>IN</sub> = 0 to V <sub>CC</sub>	-10		10	µA
I <sub>LL</sub>	Data bus 3-stage leakage current	V <sub>O</sub> = 0.4 to V <sub>CC</sub>	-10		10	µA
I <sub>X1L</sub>	X1/CLK low input current	V <sub>IN</sub> = 0, X2 grounded	-4	-2	0	mA
		V <sub>IN</sub> = 0, X2 floated	-3	-1.5	0	mA
I <sub>X1H</sub>	X1/CLK high input current	V <sub>IN</sub> = V <sub>CC</sub> , X2 grounded	-1	0.2	1	mA
		V <sub>IN</sub> = V <sub>CC</sub> , X2 floated	0	3.5	10	mA
I <sub>X2L</sub>	X2 low input current	V <sub>IN</sub> = 0, X1/CLK floated	-100	-30	0	µA
I <sub>X2H</sub>	X2 high input current	V <sub>IN</sub> = V <sub>CC</sub> , X1/CLK floated	0	+30	100	µA
I <sub>OC</sub>	Open-collector output leakage current	V <sub>O</sub> = 0.4 to V <sub>CC</sub>	-10		10	µA
I <sub>OCC</sub>	Power supply current				150	mA
	0°C to +70°C version				175	mA
	-40°C to +85°C version				175	mA

## NOTES:

- Parameters are valid over specified temperature range. See Ordering information table for applicable operating temperature range and V<sub>CC</sub> supply range.
- All voltage measurements are referenced to ground (GND). For testing, all inputs except X1/CLK swing between 0.4V and 2.4V with a transition time of 20ns maximum. For X1/CLK this swing is between 0.4V and 4.4V. All time measurements are referenced at input voltages of 0.8V and 2.0V as appropriate.
- Typical values are at +25°C, typical supply voltages, and typical processing parameters.
- T<sub>A</sub> < 0°C
- T<sub>A</sub> ≥ 0°C

## Dual asynchronous receiver/transmitter (DUART)

SCN2681

**AC CHARACTERISTICS**  $T_A = -55^{\circ}\text{C}$  to  $+125^{\circ}\text{C}$ <sup>1</sup>,  $V_{CC} = +5.0\text{V} \pm 10\%$ <sup>2, 3, 4, 5</sup>

SYMBOL	PARAMETER	LIMITS			UNIT
		Min	Typ	Max	
<b>Reset Timing (Figure 1)</b>					
t <sub>RES</sub>	RESET pulse width	200			ns
<b>Bus Timing (Figure 6)</b>					
t <sub>AS</sub>	A0-A3 setup time to RDN, WRN Low	10			ns
t <sub>AH</sub>	A0-A3 hold time from RDN, WRN Low	100			ns
t <sub>CS</sub>	CEN setup time to RDN, WRN Low	0			ns
t <sub>CH</sub>	CEN hold time from RDN, WRN High	0			ns
t <sub>RW</sub>	WRN, RDN pulse width	225			ns
t <sub>DD</sub>	Data valid after RDN Low			175	ns
t <sub>DF</sub>	Data bus floating after RDN High			100	ns
t <sub>DS</sub>	Data setup time before WRN High	100			ns
t <sub>DH</sub>	Data hold time after WRN High	20			ns
t <sub>RWD</sub>	High time between READs and/or WRITE <sup>7, 8</sup>	200			ns
<b>Port Timing (Figure 3)<sup>6</sup></b>					
t <sub>PS</sub>	Port input setup time before RDN Low	0			ns
t <sub>PH</sub>	Port input hold time after RDN High	0			ns
t <sub>PD</sub>	Port output valid after WRN High			400	ns
<b>Interrupt Timing (Figure 4)</b>					
t <sub>IR</sub>	INTRN (or OP3-OP7 when used as interrupts) negated from: Read RHR (RxRDY/FFULL interrupt) Write THR (TxRDY interrupt) Reset command (delta break interrupt) Stop C/T command (counter interrupt) Read IPCR (input port change interrupt) Write IMR (clear of interrupt mask bit)			300 300 300 300 300 300	ns ns ns ns ns ns
<b>Clock Timing (Figure 5)<sup>10</sup></b>					
t <sub>CLK</sub>	X1/CLK High or Low time	100			ns
f <sub>CLK</sub>	X1/CLK frequency	2.0	3.6864	4.0	MHz
t <sub>CTC</sub>	CTCLK (IP2) High or Low time	100			ns
f <sub>CTC</sub>	CTCLK (IP2) frequency	0		4.0	MHz
t <sub>RX</sub> <sup>9</sup>	RxC High or Low time	220			ns
f <sub>RX</sub> <sup>9</sup>	RxC frequency (16X)	0		2.0	MHz
	(1X)	0		1.0	MHz
t <sub>TX</sub> <sup>9</sup>	TxC High or Low time	220			ns
f <sub>TX</sub> <sup>9</sup>	TxC frequency (16X)	0		2.0	MHz
	(1X)	0		1.0	MHz
<b>Transmitter Timing (Figure 6)</b>					
t <sub>TXD</sub> <sup>9</sup>	TxD output delay from TxC Low			350	ns
t <sub>TCS</sub> <sup>9</sup>	Output delay from TxC Low to TxD data output	0		150	ns
<b>Receiver Timing (Figure 7)</b>					
t <sub>RXS</sub> <sup>9</sup>	RxD data setup time to RxC High	240			ns
t <sub>RXH</sub> <sup>9</sup>	RxD data hold time from RxC High	200			ns

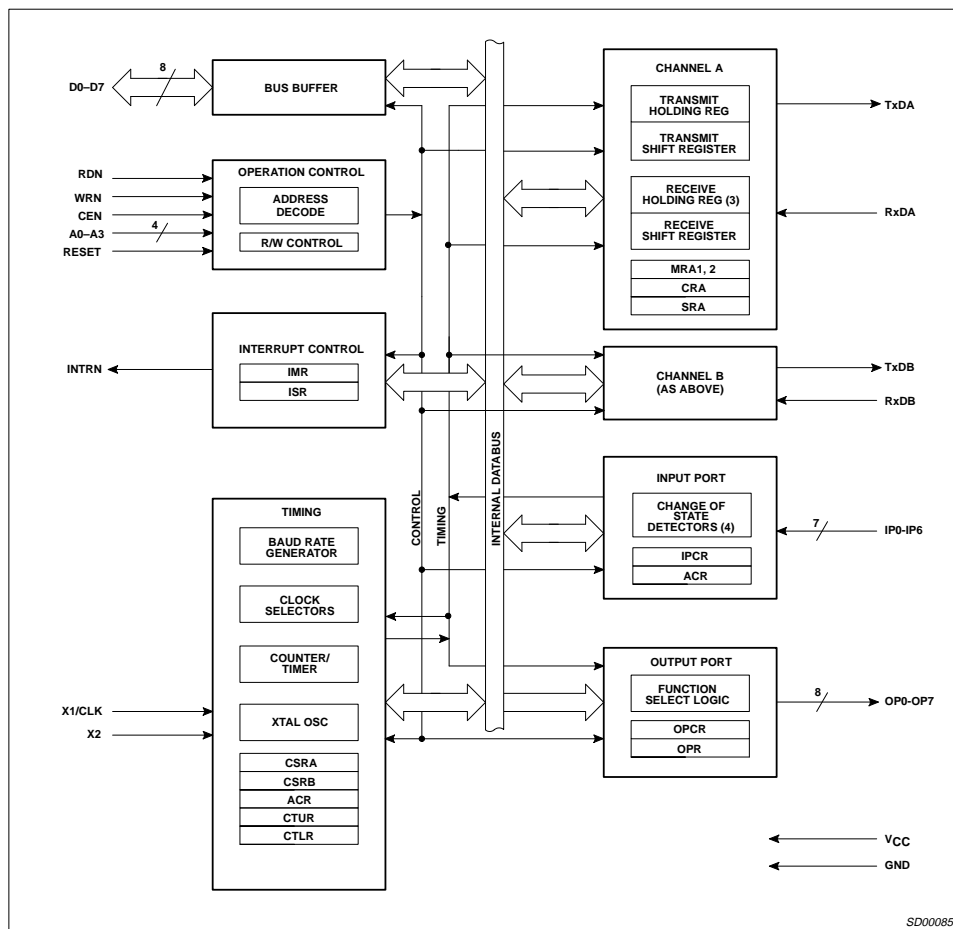
**NOTES:**

- For operating at elevated temperatures, the device must be derated based on  $+150^{\circ}\text{C}$  maximum junction temperature.
- Parameters are valid over specified temperature range.
- All voltage measurements are referenced to ground (GND). For testing, all inputs except X1/CLK swing between 0.4V and 2.4V with a transition time of  $\leq 20\text{ns}$ . For X1/CLK this swing is between 0.4V and 4.4V. All time measurements are referenced at input voltages of 0.8V and 2.0V as appropriate.
- Typical values are at  $+25^{\circ}\text{C}$ , typical supply voltages, and typical processing parameters.
- Test condition for outputs:  $C_L = 150\text{pF}$ , except interrupt outputs. Test condition for interrupt outputs:  $C_L = 50\text{pF}$ ,  $R_L = 2.7\text{k}\Omega$  to  $V_{CC}$ .
- Timing is illustrated and referenced to the WRN and RDN inputs. The device may also be operated with CEN as the 'strobing' input. In this case, all timing specifications apply referenced to the falling and rising edges of CEN. CEN and RDN (also CEN and WRN) are ANDed internally. As a consequence, the signal asserted last initiates the cycle and the signal negated first terminates the cycle.
- If CEN is used as the 'strobing' input, the parameter defines the minimum High times between one CEN and the next. The RDN signal must be negated for t<sub>RWD</sub> to guarantee that any status register changes are valid.
- Consecutive write operations to the same command register require at least three edges of the X1 clock between writes.
- This parameter is not applicable to the 28-pin device.
- Operation to 0MHz is assured by design. However, operation at low frequencies is not tested and has not been characterized.

## Dual asynchronous receiver/transmitter (DUART)

SCN2681

## BLOCK DIAGRAM



## BLOCK DIAGRAM

The SCN2681 DUART consists of the following eight major sections: data bus buffer, operation control, interrupt control, timing, communications Channels A and B, input port and output port. Refer to the block diagram.

## Data Bus Buffer

The data bus buffer provides the interface between the external and internal data buses. It is controlled by the operation control block to allow read and write operations to take place between the controlling CPU and the DUART.

## Operation Control

The operation control logic receives operation commands from the CPU and generates appropriate signals to internal sections to control device operation. It contains address decoding and read and write circuits to permit communications with the microprocessor via the data bus buffer.

## Interrupt Control

A single active-Low interrupt output (INTRN) is provided which is activated upon the occurrence of any of eight internal events. Associated with the interrupt system are the Interrupt Mask Register



## Dual asynchronous receiver/transmitter (DUART)

SCN2681

(IMR) and the Interrupt Status Register (ISR). The IMR may be programmed to select only certain conditions to cause INTRN to be asserted. The ISR can be read by the CPU to determine all currently active interrupting conditions.

Outputs OP3-OP7 can be programmed to provide discrete interrupt outputs for the transmitter, receivers, and counter/timer.

**Timing Circuits**

The timing block consists of a crystal oscillator, a baud rate generator, a programmable 16-bit counter/timer, and four clock selectors. The crystal oscillator operates directly from a 3.6864MHz crystal connected across the X1/CLK and X2 inputs. If an external clock of the appropriate frequency is available, it may be connected to X1/CLK. The clock serves as the basic timing reference for the Baud Rate Generator (BRG), the counter/timer, and other internal circuits. A clock signal within the limits specified in the specifications section of this data sheet must always be supplied to the DUART.

If an external clock is used instead of a crystal, both X1 and X2 should use a configuration similar to the one in Figure 5.

The baud rate generator operates from the oscillator or external clock input and is capable of generating 18 commonly used data communications baud rates ranging from 50 to 38.4k baud. The clock outputs from the BRG are at 16X the actual baud rate. The counter/timer can be used as a timer to produce a 16X clock for any other baud rate by counting down the crystal clock or an external clock. The four clock selectors allow the independent selection, for each receiver and transmitter, of any of these baud rates or external timing signal.

**Counter/Timer (C/T)**

The counter timer is a 16-bit programmable divider that operates in one of three modes: counter, timer, time out. In the timer mode it generates a square wave. In the counter mode it generates a time delay. In the time out mode it monitors the time between received characters. The C/T uses the numbers loaded into the Counter/Timer Lower Register (CTLR) and the Counter/Timer Upper Register (CTUR) as its divisor.

The counter timer is controlled with six commands: Start/Stop C/T, Read/Write Counter/Timer lower register and Read/Write Counter/Timer upper register. These commands have slight differences depending on the mode of operation. Please see the detail of the commands under the CTLR/CTUR Register descriptions.

**Communications Channels A and B**

Each communications channel of the SCN2681 comprises a full-duplex asynchronous receiver/transmitter (UART). The operating frequency for each receiver and transmitter can be selected independently from the baud rate generator, the counter timer, or from an external input.

The transmitter accepts parallel data from the CPU, converts it to a serial bit stream, inserts the appropriate start, stop, and optional parity bits and outputs a composite serial stream of data on the TxD output pin. The receiver accepts serial data on the RxD pin, converts this serial input to parallel format, checks for start bit, stop bit, parity bit (if any), or break condition and sends an assembled character to the CPU.

The input port pulse detection circuitry uses a 38.4kHz sampling clock derived from one of the baud rate generator taps. This results in a sampling period of slightly more than 25µs (this assumes that

the clock input is 3.6864MHz). The detection circuitry, in order to guarantee that a true change in level has occurred, requires two successive samples at the new logic level be observed. As a consequence, the minimum duration of the signal change is 25µs if the transition occurs "coincident with the first sample pulse". The 50µs time refers to the situation in which the change-of-state is "just missed" and the first change-of-state is not detected until 25µs later.

**Input Port**

The inputs to this unlatched 7-bit port can be read by the CPU by performing a read operation at address D16. A High input results in a logic 1 while a Low input results in a logic 0. D7 will always read as a logic 1. The pins of this port can also serve as auxiliary inputs to certain portions of the DUART logic.

Four change-of-state detectors are provided which are associated with inputs IP3, IP2, IP1 and IP0. A High-to-Low or Low-to-High transition of these inputs lasting longer than 25 – 50µs, will set the corresponding bit in the input port change register. The bits are cleared when the register is read by the CPU. Any change-of-state can also be programmed to generate an interrupt to the CPU.

**Output Port**

The output port pins may be controlled by the OPR, OPCR, MR and CR registers. Via appropriate programming they may be just another parallel port to external circuits, or they may represent many internal conditions of the UART. When this 8-bit port is used as a general purpose output port, the output port pins drive a state which is the complement of the Output Port Register (OPR). OPR(n) = 1 results in OP(n) = Low and vice versa. Bits of the OPR can be individually set and reset. A bit is set by performing a write operation at address E16 with the accompanying data specifying the bits to be set (1 = set, 0 = no change).

Likewise, a bit is reset by a write at address F16 with the accompanying data specifying the bits to be reset (1 = reset, 0 = no change).

Outputs can be also individually assigned specific functions by appropriate programming of the Channel A mode registers (MR1A, MR2A), the Channel B mode registers (MR1B, MR2B), and the Output Port Configuration Register (OPCR).

**TRANSMITTER OPERATION**

The SCN2681 is conditioned to transmit data when the transmitter is enabled through the command register. The SCN2681 indicates to the CPU that it is ready to accept a character by setting the TxRDY bit in the status register. This condition can be programmed to generate an interrupt request at OP6 or OP7 and INTRN. When a character is loaded into the Transmit Holding Register (THR), the above conditions are negated. Data is transferred from the holding register to transmit shift register when it is idle or has completed transmission of the previous character. The TxRDY conditions are then asserted again which means one full character time of buffering is provided. Characters cannot be loaded into the THR while the transmitter is disabled.

The transmitter converts the parallel data from the CPU to a serial bit stream on the TxD output pin. It automatically sends a start bit followed by the programmed number of data bits, an optional parity bit, and the programmed number of stop bits. The least significant bit is sent first. Following the transmission of the stop bits, if a new character is not available in the THR, the TxD output remains High and the TxEMT bit in the Status Register (SR) will be set to 1.

## Dual asynchronous receiver/transmitter (DUART)

SCN2681

Transmission resumes and the TxEMT bit is cleared when the CPU loads a new character into the THR.

If the transmitter is disabled, it continues operating until the character currently being transmitted is completely sent out. The transmitter can be forced to send a continuous Low condition by issuing a send break command.

The transmitter can be reset through a software command. If it is reset, operation ceases immediately and the transmitter must be enabled through the command register before resuming operation. If CTS operation is enable, the CTSN input must be Low in order for the character to be transmitted. If it goes High in the middle of a transmission, the character in the shift register is transmitted and TxDA then remains in the marking state until CTSN goes Low. The transmitter can also control the deactivation of the RTSN output. If programmed, the RTSN output will be reset one bit time after the character in the transmit shift register and transmit holding register (if any) are completely transmitted, if the transmitter has been disabled.

**Receiver**

The SCN2681 is conditioned to receive data when enabled through the command register. The receiver looks for a High-to-Low (mark-to-space) transition of the start bit on the RxD input pin. If a transition is detected, the state of the RxD pin is sampled each 16X clock for 7 1/2 clocks (16X clock mode) or at the next rising edge of the bit time clock (1X clock mode). If RxD is sampled High, the start bit is invalid and the search for a valid start bit begins again. If RxD is still Low, a valid start bit is assumed and the receiver continues to sample the input at one bit time intervals at the theoretical center of the bit, until the proper number of data bits and parity bit (if any) have been assembled, and one stop bit has been detected. The least significant bit is received first. The data is then transferred to the Receive Holding Register (RHR) and the RxRDY bit in the SR is set to a 1. This condition can be programmed to generate an interrupt at OP4 or OP5 and INTRN. If the character length is less than eight bits, the most significant unused bits in the RHR are set to zero.

After the stop bit is detected, the receiver will immediately look for the next start bit. However, if a non-zero character was received without a stop bit (framing error) and RxD remains Low for one half of the bit period after the stop bit was sampled, then the receiver operates as if a new start bit transition had been detected at that point (one-half bit time after the stop bit was sampled).

The parity error, framing error, overrun error and received break state (if any) are strobed into the SR at the received character boundary, before the RxRDY status bit is set. If a break condition is detected (RxD is Low for the entire character including the stop bit), a character consisting of all zeros will be loaded into the RHR and the received break bit in the SR is set to 1. The RxD input must return to high for two (2) clock edges of the X1 crystal clock for the receiver to recognize the end of the break condition and begin the search for a start bit. **This will usually require a high time of one X1 clock period or 3 X1 edges since the clock of the controller is not synchronous to the X1 clock.**

**Receiver FIFO**

The RHR consists of a First-In-First-Out (FIFO) stack with a capacity of three characters. Data is loaded from the receive shift register into the topmost empty position of the FIFO. The RxRDY bit in the status register is set whenever one or more characters are available to be read, and a FFULL status bit is set if all three stack

positions are filled with data. Either of these bits can be selected to cause an interrupt. A read of the RHR outputs the data at the top of the FIFO. After the read cycle, the data FIFO and its associated status bits (see below) are 'popped' thus emptying a FIFO position for new data.

**Receiver Status Bits**

In addition to the data word, three status bits (parity error, framing error, and received break) are also appended to each data character in the FIFO (overrun is not). Status can be provided in two ways, as programmed by the error mode control bit in the mode register. In the 'character' mode, status is provided on a character-by-character basis; the status applies only to the character at the top of the FIFO. In the 'block' mode, the status provided in the SR for these three bits is the logical-OR of the status for all characters coming to the top of the FIFO since the last 'reset error' command was issued. In either mode reading the SR does not affect the FIFO. The FIFO is 'popped' only when the RHR is read. Therefore the status register should be read prior to reading the FIFO.

If the FIFO is full when a new character is received, that character is held in the receive shift register until a FIFO position is available. If an additional character is received while this state exists, the contents of the FIFO are not affected; the character previously in the shift register is lost and the overrun error status bit (SR[4]) will be set-upon receipt of the start bit of the new (overrunning) character.

The receiver can control the deactivation of RTS. If programmed to operate in this mode, the RTSN output will be negated when a valid start bit was received and the FIFO is full. When a FIFO position becomes available, the RTSN output will be re-asserted automatically. This feature can be used to prevent an overrun, in the receiver, by connecting the RTSN output to the CTSN input of the transmitting device.

**Receiver Reset and Disable**

Receiver disable stops the receiver immediately – data being assembled if the receiver shift register is lost. Data and status in the FIFO is preserved and may be read. A re-enable of the receiver after a disable will cause the receiver to begin assembling characters at the next start bit detected.

A receiver reset will discard the present shift register data, reset the receiver ready bit (RxRDY), clear the status of the byte at the top of the FIFO and re-align the FIFO read/write pointers. This has the appearance of "clearing or flushing" the receiver FIFO. In fact, the FIFO is NEVER cleared! The data in the FIFO remains valid until overwritten by another received character. Because of this, erroneous reading or extra reads of the receiver FIFO will miss-align the FIFO pointers and result in the reading of previously read data. A receiver reset will re-align the pointers.

**Multidrop Mode**

The DUART is equipped with a wake up mode for multidrop applications. This mode is selected by programming bits MR1A[4:3] or MR1B[4:3] to '11' for Channels A and B, respectively. In this mode of operation, a 'master' station transmits an address character followed by data characters for the addressed 'slave' station. The slave stations, with receivers that are normally disabled, examine the received data stream and 'wake up' the CPU (by setting RxRDY) only upon receipt of an address character. The CPU compares the received address to its station address and enables the receiver if it wishes to receive the subsequent data characters. Upon receipt of

## Dual asynchronous receiver/transmitter (DUART)

SCN2681

another address character, the CPU may disable the receiver to initiate the process again.

A transmitted character consists of a start bit, the programmed number of data bits, and Address/Data (A/D) bit, and the programmed number of stop bits. The polarity of the transmitted A/D bit is selected by the CPU by programming bit MR1A[2]/MR1B[2]. MR1A[2]/MR1B[2] = 0 transmits a zero in the A/D bit position, which identifies the corresponding data bits as data

while MR1A[2]/MR1B[2] = 1 transmits a one in the A/D bit position, which identifies the corresponding data bits as an address. The CPU should program the mode register prior to loading the corresponding data bits into the THR.

In this mode, the receiver continuously looks at the received data stream, whether it is enabled or disabled. If disabled, it sets the RxRDY status bit and loads the character into the RHR FIFO if the

Table 1. SCN2681 Register Addressing

A3	A2	A1	A0	READ (RDN = 0)	WRITE (WRN = 0)
0	0	0	0	Mode Register A (MR1A, MR2A)	Mode Register A (MR1A, MR2A)
0	0	0	1	Status Register A (SRA)	Clock Select Register A (CSRA)
0	0	1	0	BRG Test *	Command Register A (CRA)
0	0	1	1	Rx Holding Register A (RHRA)	Tx Holding Register A (THRA)
0	1	0	0	Input Port Change Register (IPCR)	Aux. Control Register (ACR)
0	1	0	1	Interrupt Status Register (ISR)	Interrupt Mask Register (IMR)
0	1	1	0	Counter/Timer Upper (CTU)	C/T Upper Register (CRUR)
0	1	1	1	Counter/Timer Lower (CTL)	C/T Lower Register (CTLR)
1	0	0	0	Mode Register B (MR1B, MR2B)	Mode Register B (MR1B, MR2B)
1	0	0	1	Status Register B (SRB)	Clock Select Register B (CSRB)
1	0	1	0	1X/16X Test	Command Register B (CRB)
1	0	1	1	Rx Holding Register B (RHRB)	Tx Holding Register B (THRB)
1	1	0	0	*Reserved*	*Reserved*
1	1	0	1	Input Port	Output Port Conf. Register (OPCR)
1	1	1	0	Start Counter Command	Set Output Port Bits Command
1	1	1	1	Stop Counter Command	Reset Output Port Bits Command

\* See Table 5 for BRG Test frequencies in this data sheet, and "Extended baud rates for SCN2681, SCN68681, SCC2691, SCC2692, SCC68681 and SCC2698B" Philips Semiconductors ICs for Data Communications, IC-19, 1994.

received A/D bit is a one (address tag), but discards the received character if the received A/D bit is a zero (data tag). If enabled, all received characters are transferred to the CPU via the RHR. In either case, the data bits are loaded into the data FIFO while the A/D bit is loaded into the status FIFO position normally used for parity error (SRA[5] or SRB[5]). Framing error, overrun error, and break detect operate normally whether or not the receive is enabled.

### PROGRAMMING

The operation of the DUART is programmed by writing control words into the appropriate registers. Operational feedback is provided via status registers which can be read by the CPU. The addressing of the registers is described in Table 1.

The contents of certain control registers are initialized to zero on RESET. Care should be exercised if the contents of a register are changed during operation, since certain changes may cause operational problems.

For example, changing the number of bits per character while the transmitter is active may cause the transmission of an incorrect

character. In general, the contents of the MR, the CSR, and the OPCR should only be changed while the receiver(s) and transmitter(s) are not enabled, and certain changes to the ACR should only be made while the C/T is stopped.

Mode registers 1 and 2 of each channel are accessed via independent auxiliary pointers. The pointer is set to MR1x by RESET or by issuing a 'reset pointer' command via the corresponding command register. Any read or write of the mode register while the pointer is at MR1x, switches the pointer to MR2x. The pointer then remains at MR2x, so that subsequent accesses are always to MR2x unless the pointer is reset to MR1x as described above.

Mode, command, clock select, and status registers are duplicated for each channel to provide total independent operation and control. Refer to Table 2 for register bit descriptions. The reserved registers at addresses H'02' and H'0A' should never be read during normal operation since they are reserved for internal diagnostics.

## Dual asynchronous receiver/transmitter (DUART)

SCN2681

Table 2. Register Bit Formats

MR1A MR1B	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	RxRTS CONTROL	RxINT SELECT	ERROR MODE*	PARITY MODE		PARITY TYPE	BITS PER CHARACTER	
	0 = No 1 = Yes	0 = RxRDY 1 = FFULL	0 = Char 1 = Block	00 = With Parity 01 = Force Parity 10 = No Parity 11 = Multidrop Mode	0 = Even 1 = Odd	00 = 5 01 = 6 10 = 7 11 = 8		

## NOTE:

\*In block error mode, block error conditions must be cleared by using the error reset command (command 4x) or a receiver reset.

MR2A MR2B	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	CHANNEL MODE		TxRTS CONTROL	CTS ENABLE Tx	STOP BIT LENGTH*			
	00 = Normal 01 = Auto-Echo 10 = Local loop 11 = Remote loop	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = 0.563 1 = 0.625 2 = 0.688 3 = 0.750	4 = 0.813 5 = 0.875 6 = 0.938 7 = 1.000	8 = 1.563 9 = 1.625 A = 1.688 B = 1.750	C = 1.813 D = 1.875 E = 1.938 F = 2.000

## NOTE:

\*Add 0.5 to values shown for 0 – 7 if channel is programmed for 5 bits/char.

CSRA CSRB	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	RECEIVER CLOCK SELECT				TRANSMITTER CLOCK SELECT			
	See Text				See Text			

## NOTE:

\* See "Extended baud rates for SCN2681, SCN68681, SCC2691, SCC2692, SCC68681 and SCC2698B" Philips Semiconductors ICs for Data Communications, IC-19, 1994.

CRA CRB	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	MISCELLANEOUS COMMANDS				DISABLE Tx	ENABLE Tx	DISABLE Rx	ENABLE Rx
	Not used – should be 0	See Text			0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes

## NOTE:

\*Access to the upper four bits of the command register should be separated by three (3) edges of the X1 clock. A disabled transmitter cannot be loaded.

SRA SRB	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	RECEIVED BREAK*	FRAMING ERROR*	PARITY ERROR*	OVERRUN ERROR	TxEINT	TxRDY	FFULL	RxRDY
	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes

## NOTE:

\* These status bits are appended to the corresponding data character in the receive FIFO. A read of the status provides these bits (7:5) from the top of the FIFO together with bits (4:0). These bits are cleared by a "reset error status" command. In character mode they are discarded when the corresponding data character is read from the FIFO. In block error mode, block error conditions must be cleared by using the error reset command (command 4x) or a receiver reset.

OPCR	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	OP7	OP6	OP5	OP4	OP3		OP2	
	0 = OPR[7] 1 = TxRDYB	0 = OPR[6] 1 = TxRDYA	0 = OPR[5] 1 = RxRDY/ FFULLB	0 = OPR[4] 1 = RxRDY/ FFULLA	00 = OPR[3] 01 = C/T OUTPUT 10 = TxCB(1x) 11 = RxCB(1x)	00 = OPR[2] 01 = TxCA(16x) 10 = TxCA(1x) 11 = RxCA(1x)		

ACR	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	BRG SET SELECT	COUNTER/TIMER MODE AND SOURCE			DELTA IP 3 INT	DELTA IP 2 INT	DELTA IP 1 INT	DELTA IP 0 INT
	0 = set 1 1 = set 2	See Table 4			0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On

## Dual asynchronous receiver/transmitter (DUART)

SCN2681

Table 2. Register Bit Formats (Continued)

IPCR	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	DELTA IP 3	DELTA IP 2	DELTA IP 1	DELTA IP 0	IP 3	IP 2	IP 1	IP 0
	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = Low 1 = High	0 = Low 1 = High	0 = Low 1 = High	0 = Low 1 = High
ISR	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	INPUT PORT CHANGE	DELTA BREAK B	RxRDY/ FFULLB	TxRDYB	COUNTER READY	DELTA BREAK A	RxRDY/ FFULLA	TxRDYA
	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes
IMR	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	IN. PORT CHANGE INT	DELTA BREAK B INT	RxRDY/ FFULLB INT	TxRDYB INT	COUNTER READY INT	DELTA BREAK A INT	RxRDY/ FFULLA INT	TxRDYA INT
	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On
CTUR	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	C/T[15]	C/T[14]	C/T[13]	C/T[12]	C/T[11]	C/T[10]	C/T[9]	C/T[8]
CTLR	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	C/T[7]	C/T[6]	C/T[5]	C/T[4]	C/T[3]	C/T[2]	C/T[1]	C/T[0]

**MR1A – Channel A Mode Register 1**

MR1A is accessed when the Channel A MR pointer points to MR1. The pointer is set to MR1 by RESET or by a 'set pointer' command applied via CRA. After reading or writing MR1A, the pointer will point to MR2A.

**MR1A[7] – Channel A Receiver Request-to-Send Control**

This bit controls the deactivation of the RTSAN output (OP0) by the receiver. This output is normally asserted by setting OPR[0] and negated by resetting OPR[0]. MR1A[7] = 1 causes RTSAN to be negated upon receipt of a valid start bit if the Channel A FIFO is full. However, OPR[0] is not reset and RTSAN will be asserted again when an empty FIFO position is available. This feature can be used for flow control to prevent overrun in the receiver by using the RTSAN output signal to control the CTSN input of the transmitting device.

**MR1A[6] – Channel A Receiver Interrupt Select**

This bit selects either the Channel A receiver ready status (RxRDY) or the Channel A FIFO full status (FFULL) to be used for CPU interrupts. It also causes the selected bit to be output on OP4 if it is programmed as an interrupt output via the OPCR.

**MR1A[5] – Channel A Error Mode Select**

This bit select the operating mode of the three FIFOed status bits (FE, PE, received break) for Channel A. In the 'character' mode, status is provided on a character-by-character basis; the status applies only to the character at the top of the FIFO. In the 'block' mode, the status provided in the SR for these bits is the accumulation (logical-OR) of the status for all characters coming to the top of the FIFO since the last 'reset error' command for Channel A was issued.

**MR1A[4:3] – Channel A Parity Mode Select**

If 'with parity' or 'force parity' is selected a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data MR1A[4:3] + 11 selects Channel A to operate in the special multidrop mode described in the Operation section.

**MR1A[2] – Channel A Parity Type Select**

This bit selects the parity type (odd or even) if the 'with parity' mode is programmed by MR1A[4:3], and the polarity of the forced parity bit if the 'force parity' mode is programmed. It has no effect if the 'no parity' mode is programmed. In the special multidrop mode it selects the polarity of the A/D bit.

**MR1A[1:0] – Channel A Bits Per Character Select**

This field selects the number of data bits per character to be transmitted and received. The character length does not include the start, parity, and stop bits.

**MR2A – Channel A Mode Register 2**

MR2A is accessed when the Channel A MR pointer points to MR2, which occurs after any access to MR1A. Accesses to MR2A do not change the pointer.

**MR2A[7:6] – Channel A Mode Select**

Each channel of the DUART can operate in one of four modes. MR2A[7:6] = 00 is the normal mode, with the transmitter and receiver operating independently. MR2A[7:6] = 01 places the channel in the automatic echo mode, which automatically re-transmits the received data. The following conditions are true while in automatic echo mode:

1. Received data is re-clocked and retransmitted on the TxDA output.

## Dual asynchronous receiver/transmitter (DUART)

SCN2681

2. The receive clock is used for the transmitter.
3. The receiver must be enabled, but the transmitter need not be enabled.
4. The Channel A TxRDY and TxEMT status bits are inactive.
5. The received parity is checked, but is not regenerated for transmission, i.e. transmitted parity bit is as received.
6. Character framing is checked, but the stop bits are retransmitted as received.
7. A received break is echoed as received until the next valid start bit is detected.
8. CPU to receiver communication continues normally, but the CPU to transmitter link is disabled.

Two diagnostic modes can also be configured. MR2A[7:6] = 10 selects local loopback mode. In this mode:

1. The transmitter output is internally connected to the receiver input.
2. The transmit clock is used for the receiver.
3. The TxDA output is held High.
4. The RxDA input is ignored.
5. The transmitter must be enabled, but the receiver need not be enabled.
6. CPU to transmitter and receiver communications continue normally.

The second diagnostic mode is the remote loopback mode, selected by MR2A[7:6] = 11. In this mode:

1. Received data is re-clocked and re-transmitted on the TxDA output.
2. The receive clock is used for the transmitter.
3. Received data is not sent to the local CPU, and the error status conditions are inactive.
4. The received parity is not checked and is not regenerated for transmission, i.e., transmitted parity is as received.
5. The receiver must be enabled.
6. Character framing is not checked and the stop bits are retransmitted as received.
7. A received break is echoed as received until the next valid start bit is detected.

The user must exercise care when switching into and out of the various modes. The selected mode will be activated immediately upon mode selection, even if this occurs in the middle of a received or transmitted character. Likewise, if a mode is deselected the device will switch out of the mode immediately. An exception to this is switching out of autoecho or remote loopback modes: if the deselection occurs just after the receiver has sampled the stop bit (indicated in autoecho by assertion of RxRDY), and the transmitter is enabled, the transmitter will remain in autoecho mode until the entire stop has been retransmitted.

**MR2A[5] – Channel A Transmitter Request-to-Send Control**

CAUTION: When the transmitter controls the OP pin (usually used for the RTSN signal) the meaning of the pin is not RTSN at all! Rather, it signals that the transmitter has finished the transmission (i.e., end of block).

This bit allows deactivation of the RTSN output by the transmitter. This output is manually asserted and negated by the appropriate commands issued via the command register. MR2[5] set to 1 caused the RTSN to be reset automatically one bit time after the character(s) in the transmit shift register and in the THR (if any) are completely transmitted (including the programmed number of stop bits) if a previously issued transmitter disable is pending. This feature can be used to automatically terminate the transmission as follows:

1. Program the auto-reset mode: MR2[5]=1
2. Enable transmitter, if not already enabled
3. Assert RTSN via command
4. Send message
5. After the last character of the message is loaded to the THR, disable the transmitter. (If the transmitter is underrun, a special case exists. See note below.)
6. The last character will be transmitted and the RTSN will be reset one bit time after the last stop bit is sent.

NOTE: The transmitter is in an underrun condition when both the TxRDY and the TxEMT bits are set. This condition also exists immediately after the transmitter is enabled from the disabled or reset state. When using the above procedure with the transmitter in the underrun condition, the issuing of the transmitter disable must be delayed from the loading of a single, or last, character until the TxRDY becomes active again after the character is loaded.

**MR2A[4] – Channel A Clear-to-Send Control**

If this bit is 0, CTSAN has no effect on the transmitter. If this bit is a 1, the transmitter checks the state of CTSAN (IPO) each time it is ready to send a character. If IPO is asserted (Low), the character is transmitted. If it is negated (High), the TxDA output remains in the marking state and the transmission is delayed until CTSAN goes Low. Changes in CTSAN while a character is being transmitted do not affect the transmission of that character..

**MR2A[3:0] – Channel A Stop Bit Length Select**

This field programs the length of the stop bit appended to the transmitted character. Stop bit lengths of .563 TO 1 AND .563 to 2 bits. In increments of 0.625 bit, can be programmed for character lengths of 6, 7, and 8 bits. For a character lengths of 5 bits, 1.0625 to 2 stop bits can be programmed in increments of .0625 bit.

The receiver only checks for a 'mark' condition at the center of the first stop bit position (one bit time after the last data bit, or after the parity bit is enabled) in all cases.

If an external 1X clock is used for the transmitter, MR2A[3] = 0 selects one stop bit and MR2A[3] = 1 selects two stop bits to be transmitted.

**MR1B – Channel B Mode Register 1**

MR1B is accessed when the Channel B MR pointer points to MR1. The pointer is set to MR1 by RESET or by a 'set pointer' command applied via CRB. After reading or writing MR1B, the pointer will point to MR2B.

**MR2B – Channel B Mode Register 2**

MR2B is accessed when the Channel B MR pointer points to MR2, which occurs after any access to MR1B. Accesses to MR2B do not change the pointer.

The bit definitions for mode registers 1 and 2 are identical to the bit definitions for MRA and MR2A except that all control actions apply

## Dual asynchronous receiver/transmitter (DUART)

SCN2681

to the Channel B receiver and transmitter and the corresponding inputs and outputs.

**CSRA – Channel A Clock Select Register****CSRA[7:4] – Channel A Receiver Clock Select**

This field selects the baud rate clock for the Channel A receiver as follows:

CSRA[7:4]	ACR[7] = 0	Baud Rate ACR[7] = 1
0000	50	75
0001	110	110
0010	134.5	134.5
0011	200	150
0100	300	300
0101	600	600
0110	1,200	1,200
0111	1,050	2,000
1000	2,400	2,400
1001	4,800	4,800
1010	7,200	1,800
1011	9,600	9,600
1100	38.4k	19.2k
1101	Timer	Timer
1110	IP4–16X	IP4–16X
1111	IP4–1X	IP4–1X

(See also Table 5)

The receiver clock is always a 16X clock except for CSRA[7] = 1111.

**CSRA[3:0] – Channel A Transmitter Clock Select**

This field selects the baud rate clock for the Channel A transmitter. The field definition is as per CSR[7:4] except as follows:

CSRA[3:0]	ACR[7] = 0	Baud Rate ACR[7] = 1
1110	IP3–16X	IP3–16X
1111	IP3–1X	IP3–1X

The transmitter clock is always a 16X clock except for CSR[3:0] = 1111.

**CSRB – Channel B Clock Select Register****CSRB[7:4] – Channel B Receiver Clock Select**

This field selects the baud rate clock for the Channel B receiver. The field definition is as per CSRA[7:4] except as follows:

CSRB[7:4]	ACR[7] = 0	Baud Rate ACR[7] = 1
1110	IP6–16X	IP6–16X
1111	IP6–1X	IP6–1X

The receiver clock is always a 16X clock except for CSRB[7:4] = 1111.

**CSRB[3:0] – Channel B Transmitter Clock Select**

This field selects the baud rate clock for the Channel B transmitter. The field definition is as per CSRA[7:4] except as follows:

CSRB[3:0]	ACR[7] = 0	Baud Rate ACR[7] = 1
1110	IP5–16X	IP5–16X
1111	IP5–1X	IP5–1X

The transmitter clock is always a 16X clock except for CSRB[3:0] = 1111.

**CRA – Channel A Command Register**

CRA is a register used to supply commands to Channel A. Multiple commands can be specified in a single write to CRA as long as the commands are non-conflicting, e.g., the 'enable transmitter' and 'reset transmitter' commands cannot be specified in a single command word.

**CRA[7] – Not Used**

Should be set to zero for upward compatibility with newer parts.

**CRA[6:4] – Channel A Miscellaneous Command**

The encoded value of this field may be used to specify a single command as follows:

**CRA[6:4] – COMMAND**

- 000 No command.
- 001 Reset MR pointer. Causes the Channel A MR pointer to point to MR1.
- 010 Reset receiver. Resets the Channel A receiver as if a hardware reset had been applied. The receiver is disabled and the FIFO is flushed.
- 011 Reset transmitter. Resets the Channel A transmitter as if a hardware reset had been applied.
- 100 Reset error status. Clears the Channel A Received Break, Parity Error, and Overrun Error bits in the status register (SRA[7:4]). Used in character mode to clear OE status (although RB, PE and FE bits will also be cleared) and in block mode to clear all error status after a block of data has been received.
- 101 Reset Channel A break change interrupt. Causes the Channel A break detect change bit in the interrupt status register (ISR[2]) to be cleared to zero.
- 110 Start break. Forces the TxDA output Low (spacing). If the transmitter is empty the start of the break condition will be delayed up to two bit times. If the transmitter is active the break begins when transmission of the character is completed. If a character is in the THR, the start of the break will be delayed until that character, or any other loaded subsequently are transmitted. The transmitter must be enabled for this command to be accepted.
- 111 Stop break. The TxDA line will go High (marking) within two bit times. TxDA will remain High for one bit time before the next character, if any, is transmitted.

**CRA[3] – Disable Channel A Transmitter**

This command terminates transmitter operation and reset the TxDRY and TxEMT status bits. However, if a character is being transmitted or if a character is in the THR when the transmitter is disabled, the transmission of the character(s) is completed before assuming the inactive state. A disable transmitter cannot be loaded.

## Dual asynchronous receiver/transmitter (DUART)

SCN2681

**CRA[2] – Enable Channel A Transmitter**

Enables operation of the Channel A transmitter. The TxRDY status bit will be asserted.

**CRA[1] – Disable Channel A Receiver**

This command terminates operation of the receiver immediately – a character being received will be lost. The command has no effect on the receiver status bits or any other control registers. If the special multidrop mode is programmed, the receiver operates even if it is disabled. See Operation section.

**CRA[0] – Enable Channel A Receiver**

Enables operation of the Channel A receiver. If not in the special wake up mode, this also forces the receiver into the search for start-bit state.

**CRB – Channel B Command Register**

CRB is a register used to supply commands to Channel B. Multiple commands can be specified in a single write to CRB as long as the commands are non-conflicting, e.g., the 'enable transmitter' and 'reset transmitter' commands cannot be specified in a single command word.

The bit definitions for this register are identical to the bit definitions for CRA, except that all control actions apply to the Channel B receiver and transmitter and the corresponding inputs and outputs.

**SRA – Channel A Status Register****SRA[7] – Channel A Received Break**

This bit indicates that an all zero character of the programmed length has been received without a stop bit. Only a single FIFO position is occupied when a break is received: further entries to the FIFO are inhibited until the RxDA line to the marking state for at least one-half a bit time two successive edges of the internal or external 1X clock. This will usually require a high time of one X1 clock period or 3 X1 edges since the clock of the controller is not synchronous to the X1 clock.

When this bit is set, the Channel A 'change in break' bit in the ISR (ISR[2]) is set. ISR[2] is also set when the end of the break condition, as defined above, is detected.

The break detect circuitry can detect breaks that originate in the middle of a received character. However, if a break begins in the middle of a character, it must persist until at least the end of the next character time in order for it to be detected.

**SRA[6] – Channel A Framing Error**

This bit, when set, indicates that a stop bit was not detected when the corresponding data character in the FIFO was received. The stop bit check is made in the middle of the first bit position.

**SRA[5] – Channel A Parity Error**

This bit is set when the 'with parity' or 'force parity' mode is programmed and the corresponding character in the FIFO was received with incorrect parity.

In the special multidrop mode the parity error bit stores the receive A/D bit.

**SRA[4] – Channel A Overrun Error**

This bit, when set indicates that one or more characters in the received data stream have been lost. It is set upon receipt of a new character when the FIFO is full and a character is already in the receive shift register waiting for an empty FIFO position. When this occurs, the character in the receive shift register (and its break detect, parity error and framing error status, if any) is lost.

This bit is cleared by a 'reset error status' command.

**SRA[3] – Channel A Transmitter Empty (TxEMTA)**

This bit will be set when the transmitter underruns, i.e., both the TxEMT and TxRDY bits are set. This bit and TxRDY are set when the transmitter is first enabled and at any time it is re-enabled after either (a) reset, or (b) the transmitter has assumed the disabled state. It is always set after transmission of the last stop bit of a character if no character is in the THR awaiting transmission.

It is reset when the THR is loaded by the CPU, a pending transmitter disable is executed, the transmitter is reset, or the transmitter is disabled while in the underrun condition.

**SRA[2] – Channel A Transmitter Ready (TxRDYA)**

This bit, when set, indicates that the THR is empty and ready to be loaded with a character. This bit is cleared when the THR is loaded by the CPU and is set when the character is transferred to the transmit shift register. TxRDY is reset when the transmitter is disabled or reset, and is set when the transmitter is first enabled, viz., characters loaded into the THR while the transmitter is disabled will not be transmitted.

**SRA[1] – Channel A FIFO Full (FFULLA)**

This bit is set when a character is transferred from the receive shift register to the receive FIFO and the transfer causes the FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, FFULL will not be reset when the CPU reads the RHR.

**SRA[0] – Channel A Receiver Ready (RxRDYA)**

This bit indicates that a character has been received and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift to the FIFO and reset when the CPU reads the RHR, if after this read there are not more characters still in the FIFO.

**SRB – Channel B Status Register**

The bit definitions for this register are identical to the bit definitions for SRA, except that all status applies to the Channel B receiver and transmitter and the corresponding inputs and outputs.

**OPCR – Output Port Configuration Register****OPCR[7] – OP7 Output Select**

This bit programs the OP7 output to provide one of the following:

- The complement of OPR[7].
- The Channel B transmitter interrupt output which is the complement of TxRDYB. When in this mode OP7 acts as an Open-Collector output. Note that this output is not masked by the contents of the IMR.

**OPCR[6] – OP6 Output Select**

This bit programs the OP6 output to provide one of the following:

- The complement of OPR[6].
- The Channel A transmitter interrupt output which is the complement of TxRDYA. When in this mode OP6 acts as an Open-Collector output. Note that this output is not masked by the contents of the IMR.

**OPCR[5] – OP5 Output Select**

This bit programs the OP5 output to provide one of the following:

- The complement of OPR[5].
- The Channel B transmitter interrupt output which is the complement of ISR[5]. When in this mode OP5 acts as an Open-Collector



## Dual asynchronous receiver/transmitter (DUART)

SCN2681

output. Note that this output is not masked by the contents of the IMR.

**OPCR[4] – OP4 Output Select**

This field programs the OP4 output to provide one of the following:

- The complement of OPR[4].
- The Channel B transmitter interrupt output which is the complement of ISR[1]. When in this mode OP4 acts as an Open-Collector output. Note that this output is not masked by the contents of the IMR.

**OPCR[3:2] – OP3 Output Select**

This bit programs the OP3 output to provide one of the following:

- The complement of OPR[3].
- The counter/timer output, in which case OP3 acts as an Open-Collector output. In the timer mode, this output is a square wave at the programmed frequency. In the counter mode, the output remains High until terminal count is reached, at which time it goes Low. The output returns to the High state when the counter is stopped by a stop counter command. Note that this output is not masked by the contents of the IMR.
- The 1X clock for the Channel B transmitter, which is the clock that shifts the transmitted data. If data is not being transmitted, a free running 1X clock is output.
- The 1X clock for the Channel B receiver, which is the clock that samples the received data. If data is not being received, a free running 1X clock is output.

**OPCR[1:0] – OP2 Output Select**

This field programs the OP2 output to provide one of the following:

- The complement of OPR[2].
- The 16X clock for the Channel A transmitter. This is the clock selected by CSRA[3:0], and will be a 1X clock if CSRA[3:0] = 1111.
- The 1X clock for the Channel A transmitter, which is the clock that shifts the transmitted data. If data is not being transmitted, a free running 1X clock is output.
- The 1X clock for the Channel A receiver, which is the clock that samples the received data. If data is not being received, a free running 1X clock is output.

**ACR – Auxiliary Control Register****ACR[7] – Baud Rate Generator Set Select**

This bit selects one of two sets of baud rates to be generated by the BRG:

- |        |   |
|--------|---|
| Set 1: | 50, 110, 134.5, 200, 300, 600, 1.05k, 1.2k, 2.4k, 4.8k, 7.2k, 9.6k, and 38.4k baud. |
| Set 2: | 75, 110, 134.5, 150, 300, 600, 1.2k, 1.8k, 2.0k, 2.4k, 4.8k, 9.6k, and 19.2k baud.  |

The selected set of rates is available for use by the Channel A and B receivers and transmitters as described in CSRA and CSRB. Baud rate generator characteristics are given in Table 3.

**ACR[6:4] – Counter/Timer Mode And Clock Source Select**

This field selects the operating mode of the counter/timer and its clock source as shown in Table 4.

**ACR[3:0] – IP3, IP2, IP1, IP0 Change-of-State Interrupt Enable**

This field selects which bits of the input port change register (IPCR) cause the input change bit in the interrupt status register (ISR[7]) to be set. If a bit is in the 'on' state the setting of the corresponding bit in the IPCR will also result in the setting of ISR[7], which results in

the generation of an interrupt output if IMR[7] = 1. If a bit is in the 'off' state, the setting of that bit in the IPCR has no effect on ISR[7].

**IPCR – Input Port Change Register****IPCR[7:4] – IP3, IP2, IP1, IP0 Change-of-State**

These bits are set when a change-of-state, as defined in the input port section of this data sheet, occurs at the respective input pins. They are cleared when the IPCR is read by the CPU. A read of the IPCR also clears ISR[7], the input change bit in the interrupt status register. The setting of these bits can be programmed to generate an interrupt to the CPU.

**IPCR[3:0] – IP3, IP2, IP1, IP0 Current State**

These bits provide the current state of the respective inputs. The information is unlatched and reflects the state of the input pins at the time the IPCR is read.

**ISR – Interrupt Status Register**

This register provides the status of all potential interrupt sources. The contents of this register are masked by the Interrupt Mask Register (IMR). If a bit in the ISR is a '1' and the corresponding bit in the IMR is also a '1', the INTRN output will be asserted. If the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no effect on the INTRN output. Note that the IMR does not mask the reading of the ISR – the true status will be provided regardless of the contents of the IMR. The contents of this register are initialized to 00<sub>16</sub> when the DUART is reset.

**ISR[7] – Input Port Change Status**

This bit is a '1' when a change-of-state has occurred at the IP0, IP1, IP2, or IP3 inputs and that event has been selected to cause an interrupt by the programming of ACR[3:0]. The bit is cleared when the CPU reads the IPCR.

**ISR[6] – Channel B Change In Break**

This bit, when set, indicates that the Channel B receiver has detected the beginning or the end of a received break. It is reset when the CPU issues a Channel B 'reset break change interrupt' command.

**ISR[5] – Channel B Receiver Ready or FIFO Full**

The function of this bit is programmed by MR1B[6]. If programmed as receiver ready, it indicates that a character has been received in Channel B and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR. If after this read there are more characters still in the FIFO the bit will be set again after the FIFO is 'popped'. If programmed as FIFO full, it is set when a character is transferred from the receive holding register to the receive FIFO and the transfer caused the Channel B FIFO to become full; i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, the bit will be set again when the waiting character is loaded into the FIFO.

**ISR[4] – Channel B Transmitter Ready**

This bit is a duplicate of TxRDYB (SRB[2]).

**ISR[3] – Counter Ready.**

In the counter mode, this bit is set when the counter reaches terminal count and is reset when the counter is stopped by a stop counter command.

In the timer mode, this bit is set once each cycle of the generated square wave (every other time that the counter/timer reaches zero

## Dual asynchronous receiver/transmitter (DUART)

SCN2681

count). The bit is reset by a stop counter command. The command, however, does not stop the counter/timer.

**ISR[2] – Channel A Change in Break**

This bit, when set, indicates that the Channel A receiver has detected the beginning or the end of a received break. It is reset when the CPU issues a Channel A 'reset break change interrupt' command.

**ISR[1] – Channel A Receiver Ready Or FIFO Full**

The function of this bit is programmed by MR1A[6]. If programmed as receiver ready, it indicates that a character has been received in Channel A and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR. If after this read there are more characters still in the FIFO the bit will be set again after the FIFO is 'popped'. If programmed as FIFO full, it is set when a character is transferred from the receive holding register to the receive FIFO and the transfer caused the Channel A FIFO to become full; i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, the bit will be set again when the ISR[0] and IMR waiting character is loaded into the FIFO.

**ISR[0] – Channel A Transmitter Ready**

This bit is a duplicate of TxRDYA (SRA[2]).

**IMR – Interrupt Mask Register**

The programming of this register selects which bits in the ISR causes an interrupt output. If a bit in the ISR is a '1' and the corresponding bit in the IMR is also a '1' the INTRN output will be asserted. If the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no effect on the INTRN output. Note that the IMR does not mask the programmable interrupt outputs OP3–OP7 or the reading of the ISR.

**CTUR and CTLR – Counter/Timer Registers**

The CTUR and CTLR hold the eight MSBs and eight LSBs, respectively, of the value to be used by the counter/timer in either the counter or timer modes of operation. The minimum value which may be loaded into the CTUR/CTLR registers is 0002<sub>16</sub>. Note that these registers are write-only and cannot be read by the CPU.

In the timer (programmable divider) mode, the CT generates a square wave with a period of twice the value (in clock periods) of the CTUR and CTLR.

If the value in CTUR and CTLR is changed, the current half-period will not be affected, but subsequent half periods will be. In this

mode the C/T runs continuously. Receipt of a start counter command (read with A3-A0 = 1110) causes the counter to terminate the current timing cycle and to begin a new cycle using the values in CTUR and CTLR. The waveform so generated is often used for a data clock. The formula for calculating the divisor n to load to the CTUR and CTLR for a particular 1X data clock is shown below:

$$n = \text{C/T Clock Frequency divided by } 2 \times 16 \times \text{Baud rate desired}$$

Often this division will result in a non-integer number; 26.3, for example. One can only program integer numbers in a digital divider. Therefore, 26 would be chosen. This gives a baud rate error of 0.3/26.3 which is 1.14%; well within the ability asynchronous mode of operation.

The counter ready status bit (ISR[3]) is set once each cycle of the square wave. The bit is reset by a stop counter command (read with A3-A0 = 1111). The command however, does not stop the C/T. The generated square wave is output on OP3 if it is programmed to be the C/T output.

On power up and after reset, the timer/counter runs in timer mode and can only be restarted. Because it cannot be shut off or stopped, and runs continuously in timer mode, it is recommended that at initialization, the output port (OP3) should be masked off through the OPCR[3:2] = 00 until the T/C is programmed to the desired operational state.

In the counter mode, the C/T counts down the number of pulses loaded into CTUR and CTLR by the CPU. Counting begins upon receipt of a counter command. Upon reaching terminal count (0000<sub>16</sub>), the counter ready interrupt bit (ISR[3]) is set. The counter continues counting past the terminal count until stopped by the CPU. If OP3 is programmed to be the output of the C/T, the output remains High until terminal count is reached, at which time it goes Low. The output returns to the High state and ISR[3] is cleared when the counter is stopped by a stop counter command. The CPU may change the values of CTUR and CTLR at any time, but the new count becomes effective only on the next start counter command. If new values have not been loaded, the previous count values are preserved and used for the next count cycle.

In the counter mode, the current value of the upper and lower 8 bits of the counter (CTU, CTL) may be read by the CPU.

It is recommended that the counter be stopped when reading to prevent potential problems which may occur if a carry from the lower 8 bits to the upper 8 bits occurs between the times that both halves of the counter are read. However, note that a subsequent start counter command will cause the counter to begin a new count cycle using the values in CTUR and CTLR.

## Dual asynchronous receiver/transmitter (DUART)

SCN2681

**Table 3. Bit Rate Generator Characteristics Crystal or Clock = 3.6864MHz**

NORMAL RATE (BAUD)	ACTUAL 16x CLOCK (kHz)	ERROR (%)
50	0.8	0
75	1.2	0
110	1.759	-0.069
134.5	2.153	0.059
150	2.4	0
200	3.2	0
300	4.8	0
600	9.6	0
1050	16.756	-0.260
1200	19.2	0
1800	28.8	0
2000	32.056	0.175
2400	38.4	0
4800	76.8	0
7200	115.2	0
9600	153.6	0
19.2k	307.2	0
38.4k	614.4	0

NOTE: Duty cycle of 16x clock is 50% ± 1%.

**Table 4. ACR 6:4 Field Definition**

ACR 6:4	MODE	CLOCK SOURCE
000	Counter	External (IP2)
001	Counter	TxCA – 1x clock of Channel A transmitter
010	Counter	TxCB – 1x clock of Channel B transmitter
011	Counter	Crystal or external clock (x1/CLK) divided by 16
100	Timer	External (IP2)
101	Timer	External (IP2) divided by 16
110	Timer	Crystal or external clock (x1/CLK)
111	Timer	Crystal or external clock (x1/CLK) divided by 16

NOTE: Timer mode generates a squarewave.

Dual asynchronous receiver/transmitter (DUART)

SCN2681

TIMING DIAGRAMS

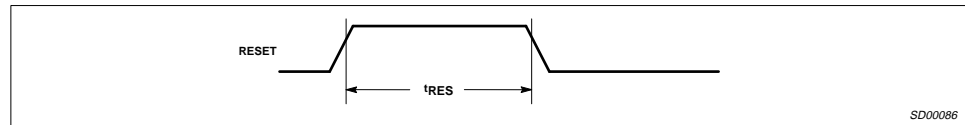


Figure 1. Reset Timing

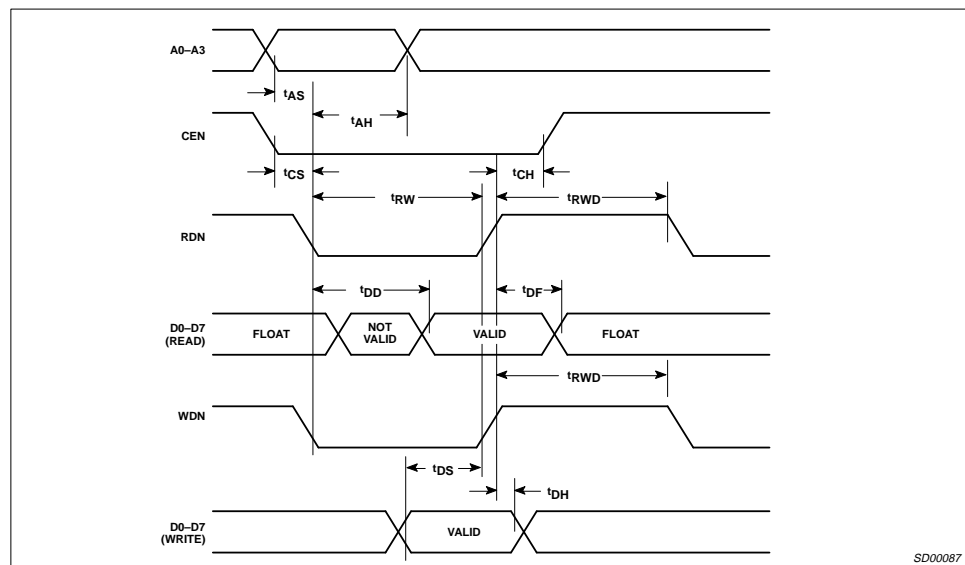


Figure 2. Bus Timing

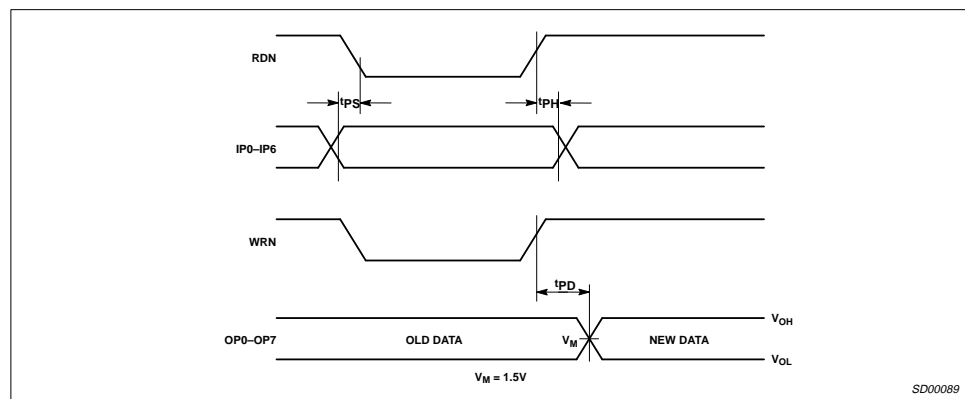


Figure 3. Port Timing

Dual asynchronous receiver/transmitter (DUART)

SCN2681

TIMING DIAGRAMS (Continued)

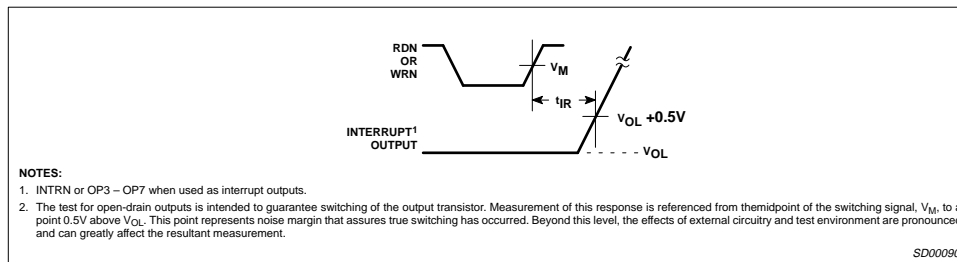


Figure 4. Interrupt Timing

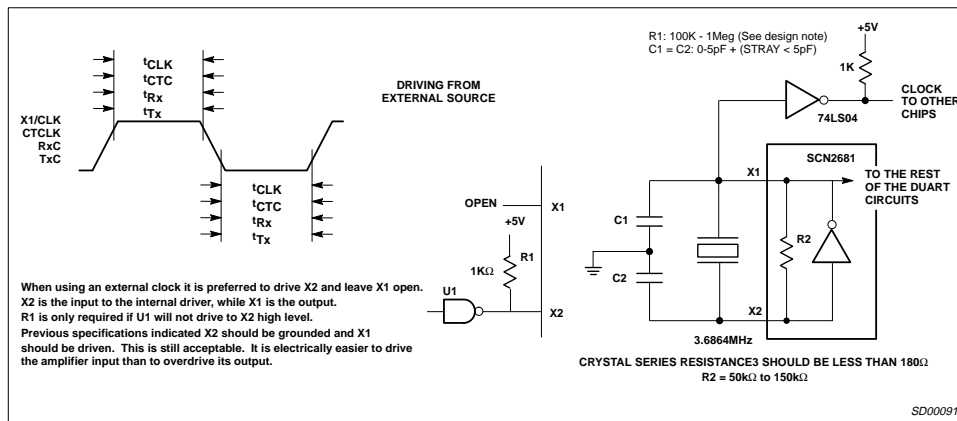


Figure 5. Clock Timing

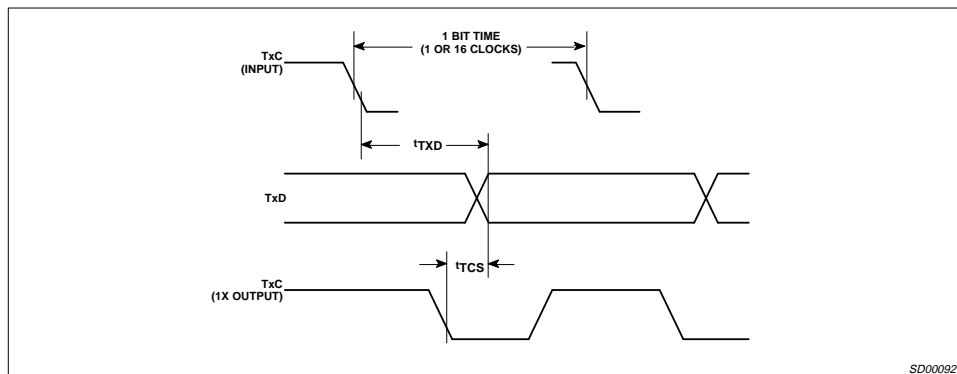


Figure 6. Transmit

Dual asynchronous receiver/transmitter (DUART)

SCN2681

TIMING DIAGRAMS (Continued)

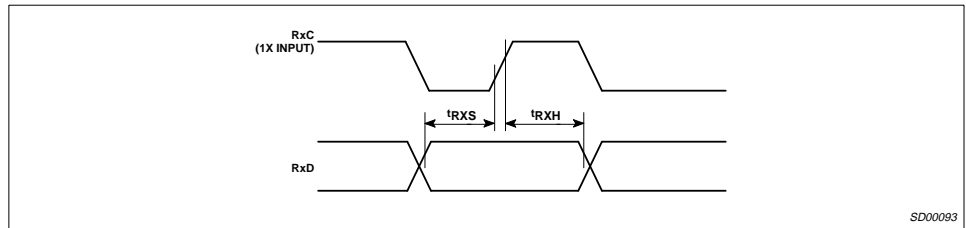
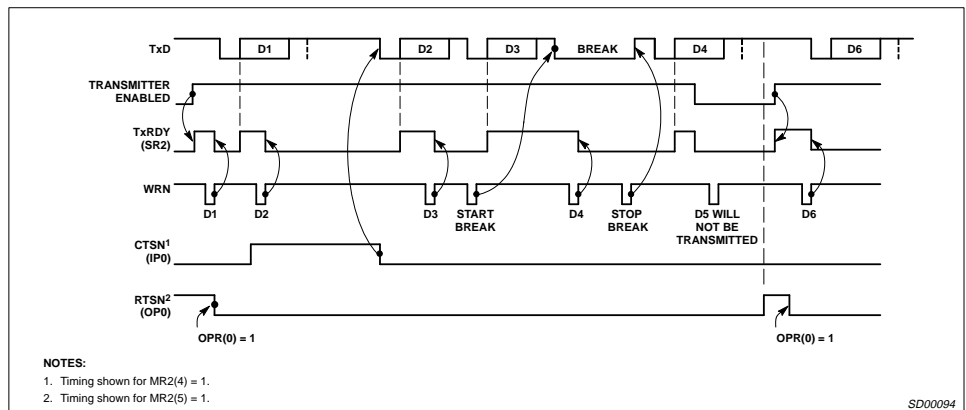


Figure 7. Receive

SD00093



- NOTES:  
 1. Timing shown for MR2(4) = 1.  
 2. Timing shown for MR2(5) = 1.

Figure 8. Transmitter Timing

SD00094

Dual asynchronous receiver/transmitter (DUART)

SCN2681

TIMING DIAGRAMS (Continued)

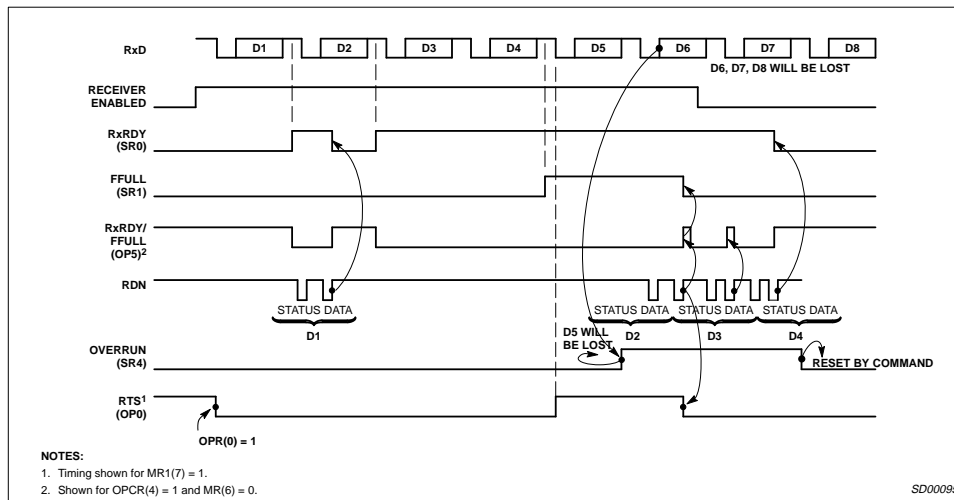


Figure 9. Receiver Timing

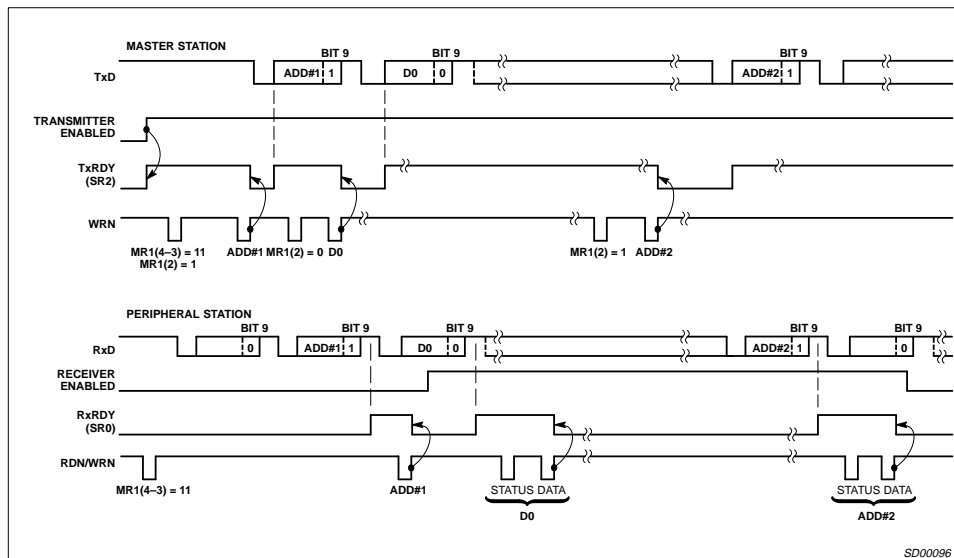


Figure 10. Wake-Up Mode

## Dual asynchronous receiver/transmitter (DUART)

SCN2681

**Output Port Notes**

The output ports are controlled from three places: the OPCR register, the OPR register, and the MR registers. The OPCR register controls the source of the data for the output ports OP2 through OP7. The data source for output ports OP0 and OP1 is controlled by the MR and CR registers. When the OPR is the source of the data for the output ports, the data at the ports is inverted from that in the OPR register. The content of the OPR register is controlled by the "Set Output Port Bits Command" and the "Reset Output Bits Command". These commands are at E and F, respectively. When these commands are used, action takes place only at the bit locations where ones exist. For example, a one in bit location 5 of the data word used with the "Set Output Port Bits" command will result in OPR5 being set to one. The OP5 would then be set to zero ( $V_{SS}$ ). Similarly, a one in bit position 5 of the data word associated with the "Reset Output Ports Bits" command would set OPR5 to zero and, hence, the pin OP5 to a one ( $V_{DD}$ ).

**The CTS, RTS, CTS Enable Tx signals**

CTS (Clear To Send) is usually meant to be a signal to the transmitter meaning that it may transmit data to the receiver. The CTS input is on pin IP0 for TxA and on IP1 for TxB. The CTS signal is active low; thus, it is called CTSAN for TxA and CTSBN for TxB.

RTS is usually meant to be a signal from the receiver indicating that the receiver is ready to receive data. It is also active low and is, thus, called RTSAN for RxA and RTSBN for RxB. RTSAN is on pin OP0 and RTSBN is on OP1. A receiver's RTS output will usually be connected to the CTS input of the associated transmitter. Therefore, one could say that RTS and CTS are different ends of the same wire!

MR2(4) is the bit that allows the transmitter to be controlled by the CTS pin (IP0 or IP1). When this bit is set to one AND the CTS input is driven high, the transmitter will stop sending data at the end of the present character being serialized. It is usually the RTS output of the receiver that will be connected to the transmitter's CTS input. The receiver will set RTS high when the receiver FIFO is full AND the start bit of the fourth character is sensed. Transmission then stops with four valid characters in the receiver. When MR2(4) is set to one, CTSN must be at zero for the transmitter to operate. If

MR2(4) is set to zero, the IP pin will have no effect on the operation of the transmitter.

MR1(7) is the bit that allows the receiver to control OP0. When OP0 (or OP1) is controlled by the receiver, the meaning of that pin will be RTS. However, a point of confusion arises in that OP0 (or OP1) may also be controlled by the transmitter. When the transmitter is controlling this pin, its meaning is not RTS at all. It is, rather, that the transmitter has finished sending its last data byte. Programming the OP0 or OP1 pin to be controlled by the receiver and the transmitter at the same time is allowed, but would usually be incompatible.

RTS is expressed at the OP0 or OP1 pin which is still an output port. Therefore, the state of OP0 or OP1 should be set low for the receiver to generate the proper RTS signal. The logic at the output is basically a NAND of the OPR register and the RTS signal as generated by the receiver. When the RTS flow control is selected via the MR(7) bit state of the OPR register is not changed. Terminating the use of "Flow Control" (via the MR registers) will return the OP0 or OP1 pins to the control of the OPR register.

**Transmitter Disable Note**

The sequence of instructions enable transmitter — load transmit holding register — disable transmitter will result in nothing being sent if the time between the end of loading the transmit holding register and the disable command is less than 3/16 bit time in the 16x mode or one bit time in the 1x mode. Also, if the transmitter, while in the enabled state and underrun condition, is immediately disabled after a single character is loaded to the transmit holding register, that character will not be sent.

In general, when it is desired to disable the transmitter before the last character is sent AND the TxEMT bit is set in the status register (TxEMT is always set if the transmitter has underrun or has just been enabled), be sure the TxRDY bit is active immediately before issuing the transmitter disable instruction. TxRDY sets at the end of the "start bit" time. It is during the start bit that the data in the transmit holding register is transferred to the transmit shift register.

Non-standard baud rates are available as shown in Table [5] below, via the BRG Test function.



## Dual asynchronous receiver/transmitter (DUART)

SCN2681

Table 5. Baud Rates Extended

CSR[7:4]	Normal BRG		BRG Test	
	ACR[7] = 0	ACR[7] = 1	ACR[7] = 0	ACR[7] = 1
0000	50	75	4,800	7,200
0001	110	110	880	880
0010	134.5	134.5	1,076	1,076
0011	200	150	19.2K	14.4K
0100	300	300	28.8K	28.8K
0101	600	600	57.6K	57.6K
0110	1,200	1,200	115.2K	115.2K
0111	1,050	2,000	1,050	2,000
1000	2,400	2,400	57.6K	57.6K
1001	4,800	4,800	4,800	4,800
1010	7,200	1,800	57.6K	14.4K
1011	9,600	9,600	9,600	9,600
1100	38.4K	19.2K	38.4K	19.2K
1101	Timer	Timer	Timer	Timer
1110	I/O2 – 16X	I/O2 – 16X	I/O2 – 16X	I/O2 – 16X
1111	I/O2 – 1X	I/O2 – 1X	I/O2 – 1X	I/O2 – 1X

NOTE: Each read on address H'2' will toggle the baud rate test mode. When in the BRG test mode, the baud rates change as shown to the left. This change affects all receivers and transmitters on the DUART.

The test mode at address H'A' changes all transmitters and receivers to the 1x mode and connects the output ports to some internal nodes.

A condition that occurs infrequently has been observed where the receiver will ignore all data. It is caused by a corruption of the start bit generally due to noise. When this occurs the receiver will appear to be asleep or locked up. The receiver must be reset for the UART to continue to function properly.

#### Reset in the Normal Mode (Receiver Enabled)

Recovery can be accomplished easily by issuing a receiver software reset followed by a receiver enable. All receiver data, status and programming will be preserved and available before reset. The reset will NOT affect the programming.

#### Reset in the Wake-Up Mode (MR1[4:3] = 11)

Recovery can also be accomplished easily by first exiting the wake-up mode (MR1[4:3] = 00 or 01 or 10), then issuing a receiver software reset followed by a wake-up re-entry (MR1[4:3] = 11). All receiver data, status and programming will be preserved and available before reset. The reset will NOT affect the programming.

The receiver has a digital filter designed to reject "noisy" data transitions and the receiver state machine was designed to reject noisy start bits or noise that might be considered a start bit. In spite of these precautions, corruption of the start bit can occur in 15ns window approximately 100ns prior to the rising edge of the data clock. The probability of this occurring is less than  $10^{-5}$  at 9600 baud.

A corrupted start bit may have some deleterious effects in ASYNC operation if it occurs within a normal data block. The receiver will tend to align its data clock to the next '0' bit in the data stream, thus potentially corrupting the remainder of the data block. A good design practice, in environments where start bit corruption is possible, is to monitor data quality (framing error, parity error, break change and received break) and "data stopped" time out periods. Time out periods can be enabled using the counter/timer in the SCC2691, SCC2692, SCC2698B and SC68692 products. This monitoring can indicate a potential start bit corruption problem.

SD00097



## C.4 CEM3360 Dual VCA





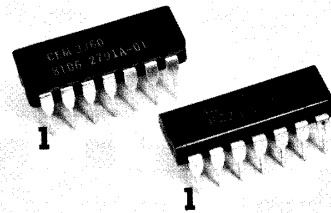
CURTIS ELECTROMUSIC SPECIALTIES

# CEM 3360

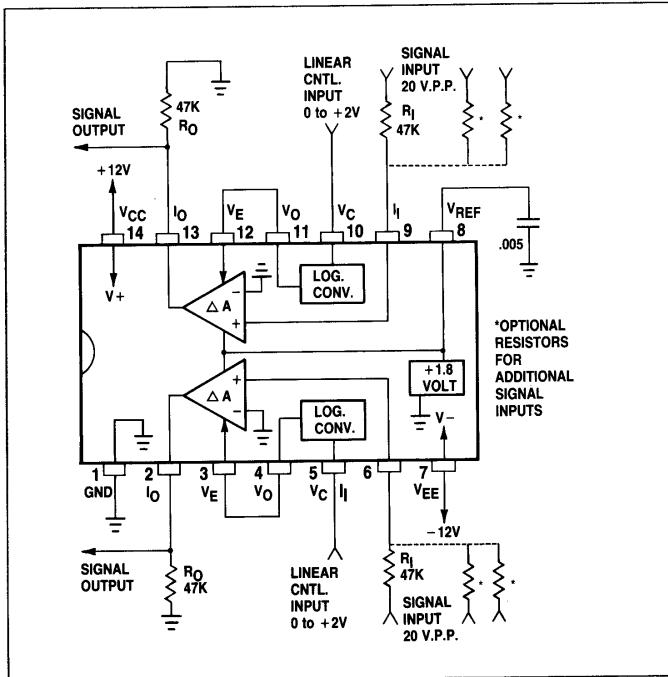
## Dual Voltage Controlled Amplifier

The CEM 3360 is a dual general purpose voltage controlled transconductor intended for such applications as voltage controlled amplifiers, filters, and waveform generators. Each transconductor independently provides both linear and exponential control scaling over greater than a 100 dB range. Complete with virtual ground summing inputs, wide voltage compliance current outputs, and control inputs referenced to ground, the CEM

3360 requires exceptionally few external components and is extremely easy to use. Because of its inherent ultra-low control feedthrough, no trimming is required. Added to these features are exceptionally low noise, wide bandwidth, and operation down to  $\pm 3$  volts, making the CEM 3360 a real cost saver in most applications requiring variable transconductance amplifiers.



### Block and Connections Diagram



#### Features:

- Low Cost
- Two Independent VCAs in a Single 14 Pin DIP
- Simple to Use - Few External Components Required
- Exceptionally Low Control Feedthrough Without Trimming: 10mV Maximum Out of 10 V.P.P. Output
- Low Noise: -110 dB Typical
- No Trimming Required
- Summing Node Signal Inputs
- Current Outputs Capable of Swinging to Within 1.5V of Each Supply
- Linear and Exponential Control
- Control Voltages Referenced to Ground
- Wide Supply Range:  $\pm 3$  to  $\pm 12$  V or  $+15$ ,  $-3$  to  $-9$  V

# CEM 3360

## Electrical Characteristics

## Notes

V <sub>CC</sub> = +12V V <sub>EE</sub> = -12V T <sub>A</sub> = 20°C				
Parameter	Minimum	Typical	Maximum	Units
Control Range, Linear and Exponential	100	—	—	dB
Control Scale Factor				mV/dB
Exponential <sup>1</sup>	+2.7	+3.0	+3.3	
Linear	48	52	56	%/V
Tempco of Control Scales				ppm
Exponential	+3000	+3300	+3600	
Linear	—	±250	±750	ppm
Control Scale Error				dB
Exponential <sup>2</sup>	—	0.6	2	
Linear	—	3.0	6.0	%
Maximum Cell Current Gain <sup>3</sup>	0.9	1.0	1.1	
Maximum Signal Input and Output Current	±300	±400	±500	μA
Signal Input Offset	-10	0	+10	mV
Control Feedthrough Without Trim <sup>4</sup>	—	±0.07	±0.3	μA
Total Harmonic Distortion <sup>3</sup>	—	1.0	3.0	%
Output Noise Current <sup>5</sup>	—	0.4	1.2	nA.R.M.S.
Signal Current Bandwidth	2.0	5.0	—	MHz
Signal Current Slew Rate <sup>3</sup>	0.5	1.5	—	mA/μS
Crosstalk Between VCAs <sup>6</sup>	-80	-90	—	dB
Signal Attenuation for Linear Control Input = 0V <sup>7</sup>	70	80	—	dB
Linear Control Voltage for Maximum Gain	1.79	1.93	2.08	V
Exponential Control Voltage Range, Referred to V <sub>REF</sub> (Pin 8)	+20	—	-280	mV
Control Input Bias Current				μA
Exponential <sup>3</sup>	-0.3	-0.8	-1.5	
Linear	-0.5	-1.6	-4.0	μA
Output Impedance <sup>3</sup>	5	12	—	Mohm
Output Voltage Compliance <sup>3</sup>	V <sub>EE</sub> + 1.2	—	V <sub>CC</sub> - 0.8	V
Reference Voltage (Pin 8)	1.7	1.8	1.9	V
Positive Supply Voltage Range <sup>8</sup>	+3.0	—	+16	V
Negative Supply Voltage Range <sup>8</sup>	-3.0	—	-16	V
Supply Current	3.8	4.8	6.0	mA

- Note 1.** Current gain is -20dB to -80dB. Control voltage is referenced to pin 8.
- Note 2.** Best straight line. Most of this error occurs at range extremities. See Hints.
- Note 3.** Output Signal Current is ±100μA.
- Note 4.** Over entire control range. Signal input is open.
- Note 5.** In 16 to 16KHz bandwidth.
- Note 6.** AT 1KHz.
- Note 7.** For negative supply less than 12 volts, this attenuation is greater. See Hints.
- Note 8.** Total supply voltage across chip should not exceed 26V.

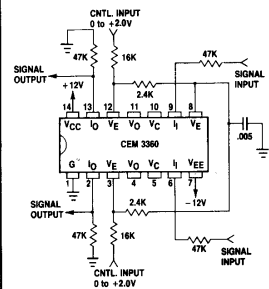
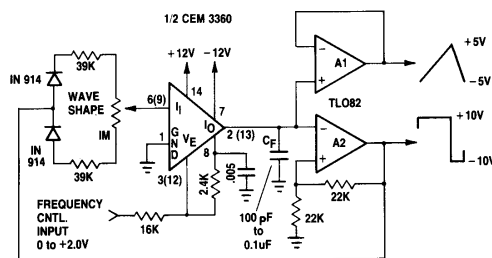
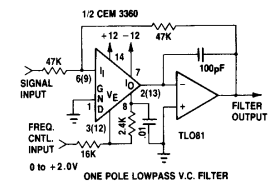


FIGURE 8: CONNECTION FOR EXPONENTIAL CONTROL SCALE



V.C. WAVEFORM GENERATOR



ONE POLE LOWPASS V.C. FILTER



Curtis Electromusic Specialties (CES) assumes no responsibility for use of any circuitry described. No circuit licenses are implied. CES reserves the right, at any time without notice, to change said circuitry. Printed U.S.A. © 1984

## **C.5 CEM3379 Analog Signal Processor**







## CEM3378/3379 Voltage Controlled Signal Processors

---

The CEM3378 and CEM3379 contain general purpose audio signal processing blocks which are completely separate from each other. These devices are useful in a wide variety of audio and musical instrument applications.

The CEM3378 includes a two-channel voltage controlled mixer, a wide range four-pole low-pass voltage controlled filter with voltage controlled resonance, and a high quality voltage controlled amplifier featuring low noise and low control voltage feedthrough without trimming.

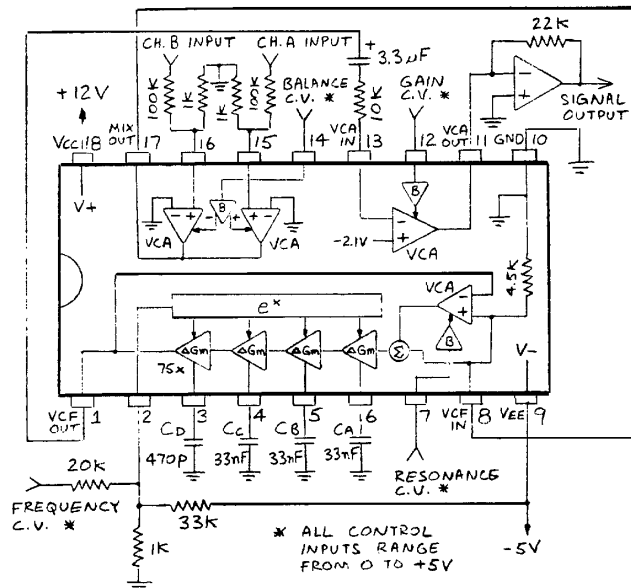
The CEM3379 includes the same filter and VCA as the CEM3378, but instead of an input mixer, provides a two-channel voltage controlled output pan function.

Since all blocks have separate input, output, and control terminals they may be interconnected as shown in the block diagrams, or used separately in different parts of a system. With the exception of the filter frequency, all control voltage inputs range from 0 to +5V and provide moderately high impedance for minimal system loading; the filter frequency control voltage ranges from -150mV to +100mV, allowing easy control voltage mixing and all parameters to be conveniently controlled with a single polarity DAC.

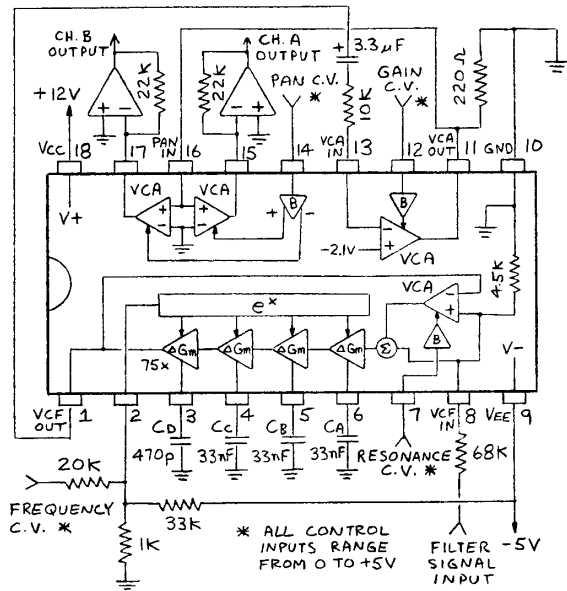
Able to operate over a wide supply range and requiring a bare minimum of external components, the CEM3378 and CEM3379 offer the designer means to create unique signal processing configurations at the lowest possible cost.

### FEATURES

- Low Cost
- VCF and 4 VCAs on a single 18 pin DIP
- Separate inputs and outputs for each function
- Choice of Balance (3378) or Pan (3379)
- Rich Sounding VCF
- Constant Amplitude versus Resonance VCF Design
- Low Noise, Low Distortion VCA
- Very Low Control Voltage Feedthrough without trims
- Operation down to +-5V



CEM 3378 BLOCK & TYPICAL CONNECTION DIAGRAM



CEM3379 BLOCK & TYPICAL CONNECTION DIAGRAM

## CEM3378/79 Electrical Characteristics

PARAMETER	MINIMUM	TYPICAL	MAXIMUM	UNIT
<b>INPUT MIXER/OUTPUT VCAs</b>				
Gain range for 0-+5V control	0-3.0	0-3.8	0.4.8	mmho
Input signal for 5% THD	---	75	---	mV pp
Attenuation at VBAL=0 or VBAL=5	80	100	120	dB
DC Control voltage feedthrough	---	10	30	uA
Signal Input Bias Current	-0.2	-0.6	-2.0	uA
Balance Control Input Bias	-1.5	-5	-15	uA
Maximum Output Current	+150	+200	+260	uA
Gain Variation (part to part)	---	0.7	2.0	dB
<b>VC FILTER</b>				
Input signal for 1% THD	---	360	---	mV PP
Passband Signal Gain Vres=0V	6.8	7.5	8.3	
Input Resistance	3.6	4.5	5.6	KOhm
Frequency Control Range	14	---	---	octaves
Frequency Control Voltage	-155	---	110	mV
Frequency Control Scale	17.5	19.0	20.5	mV/octave
Exponential Scale Error	---	0.3	1.0	%
Initial Frequency (Ca-Cc=0.033uf)	650	1000	1650	Hz
Frequency Control Input Bias	-0.2	-0.6	-2.0	uA
Resonance Control Range	Q = 0dB	---	self-osc	
Resonance Control Voltage @osc	2.2	2.8	3.4	V
Resonance Control Input Bias	-0.2	-0.5	-1.5	uA/V
DC Output Shift over 10 Octaves	---	100	250	mV pp
Output noise	---	90	---	uVrms
Maximum Output Swing	4.5	5.0	5.5	Vpp
Quiescent DC Output Voltage	-0.5	0.0	0.5	V
Output Sink Current	-0.4	-0.5	-0.6	ma
Output Source Drive Current	---	---	3.0	ma
<b>FINAL VCA</b>				
Gain Control Range	90	120	---	dB
Maximum Signal Current Gain	0.80	0.93	1.10	
Control Voltage for Max Gain	4.5	5.0	5.5	V
Control Voltage for Min Gain	30	85	140	mV
Control Input Bias Current	-0.1	-0.3	-1.0	uA/V
Voltage at Signal Input Node	-2.3	-2.1	-1.9	V
Output Voltage Range	-0.8	---	Vcc-1	V
Maximum Input Signal Swing	-200	---	200	uA
Output Noise	---	---	2.0	nA rms
THD @ +-200uA Input Swing	---	0.5	1.5	%
DC Output Offset at Min Gain	---	---	1.0	nA
DC Output Offset Range	---	+0.2	+1.2	uA
<b>GENERAL</b>				
Supply Voltage Range	+4.75	+9	+12.5	VDC

Supply Current per Chip	5.8	7.3	9.1	ma
-------------------------	-----	-----	-----	----

**POWER SUPPLIES**

The maximum supply allowed across either device is 25 volts. Due to internal voltage regulators, the supplies do not have to be balanced: +5/-12 is allowed, as would be +12/-5. Since the maximum positive output swing of the filter is 2.9 volts below the positive supply, some loss in maximum VCF output will occur at +4.75 volt supply. For best performance with low power dissipation, use +9/-5 or +12/-5 supply voltages.

**INPUT MIXER (3378) and OUTPUT PAN (3379)**

These VCAs are simple 3080 types with the inverting inputs internally connected to ground. Thus, the external input should be driven from a low impedance (<1K) referenced to ground. Control feedthrough may be trimmed if desired by applying a +5mV adjustable voltage to the input pin. Note that in the 3379, the inputs are common and so there will be a slight mismatch between sections.

The gains of the two VCAs are complementary, being equal and half of maximum at a control voltage of 2.5 volts. The control scales a linear between 1.0 and 3.5V, becoming logarithmic beyond these extremes.

Since the VCA output(s) have a limited negative output voltage compliance (-0.2V), they must be fed to a virtual ground summing node on an op amp for large output voltage swings. However, in cases where the output(s) drive another 3080-type VCA or the input of the VCF section (where the control voltage swing is less than +200mV), the output current may be converted to the required voltage simply with a resistor connected from the output pin to ground.

In the case of driving the VCF input, an external load resistor is not required since there is an internal 4.5K (nominal) resistor to ground on the VCF input pin. The VCA voltage gain from input to output is  $G_m \times R_I = 3.8 \text{ mmho} \times 4500 = 17$ . Thus, the nominal filter input of 360 mvpp is achieved with a VCA input of only 21 mvpp, allowing a typical THD of <0.1%. If more distortion can be tolerated, then a better signal-to-noise ratio can be obtained through the VCA by adding a resistor from the VCA output/VCF input to ground. A value of 1.6K for instance will lower the gain from 17 to 4.5, requiring a VCA input of 80 mvpp, or a 12dB SNR improvement.

**FILTER**

The voltage controlled filter (VCF) is the standard musical 4 pole low pass with internal feedback through a VCA to add resonance or sustained oscillation at the cut-off frequency. A portion of the input signal is applied to the resonance VCA, so that as the amount of resonance is increased, the passband gain drops by only 6dB instead of the normal 12dB without this technique. This choice of a 6dB drop ensures the peak-to-peak output level remains the same when the output waveform rings from added resonance.

If the VCF input signal comes from a source other than the mixer output, it will most likely require attenuation down to the nominal 360mv pp level. This is easily accomplished with a single series resistor to the input pin (Pin 8). The amount of attenuation is given by:

$$1 + (R_{in}/4500)$$

However, the internal 4500 ohm resistor has a 25% tolerance, so a chip-to-chip  $\pm 2.5\text{dB}$  variation is to be expected. Lower variation can be obtained by adding a shunt resistor to ground. A 1.3K shunt resistor will reduce the input resistance to 1K and the output variation caused by the 4.5K will be reduced to  $\pm 0.5\text{dB}$ . For best performance, the signal applied to the filter input should have  $< 50\text{mv}$  DC component.

The cut-off frequency of the filter (which is defined as the oscillation frequency at maximum resonance or the  $-9\text{dB}$  point at no resonance) is determined by the transconductance and associated capacitance of each of the 4 stages as:

$$f_c = G_m / (2 \times \pi \times C)$$

Since the transconductance of the last stage is  $1/75^{\text{th}}$  of the other 3 stages, the capacitor value is  $1/75^{\text{th}}$  of the other capacitors. Best sweep performance is obtained over a transconductance range of  $1\text{umho}$  to  $4\text{mmho}$ . For a desired frequency range of  $5\text{Hz}$  to  $20\text{KHz}$ , Ca, Cb, and Cc are chosen to be  $33\text{nF}$  and Cd becomes  $470\text{pF}$ . Note that the frequency can be swept one octave above and below these frequencies.

The transconductance is varied in an exponential manner with the control voltage, and is given in umhos by

$$G_m = 200 \exp(V_{\text{freq}}/V_T)$$

where  $V_T$  is approximately  $28.5\text{mv}$  at  $20\text{C}$  and has a temperature coefficient of  $+3300\text{ppm}$ . Note that when  $V_{\text{freq}} = 0$ , the transconductance is nominally  $200\text{umho}$ , resulting in a cutoff frequency of around  $1\text{KHz}$  with the capacitors given. The lower frequency of  $5\text{Hz}$  is 7.6 octaves below the zero control voltage. This requires a  $-150\text{mv}$  signal. The upper limit of  $20\text{KHz}$  requires a  $90\text{mv}$  control voltage signal.

In the usual case, the system frequency control voltage must be attenuated with a resistor divider down to these levels. If the system CV ranges from 0 to a positive value (most likely), then an additional resistor between the control pin and the negative supply voltage is needed to produce a negative voltage for the lower cutoff frequencies. For best results, the input impedance to the control pin should be  $< 2\text{K}$ . Although the transconductors themselves have been internally temperature compensated, the control scale still has a  $-3300\text{ppm}$  factor due to TC. Therefore, a  $+3300\text{ppm}$  temperature compensation resistor is used in the CV attenuation network.

The VCF output (Pin 1) is a low impedance output capable of driving loads down to  $6.8\text{K}$ . If more drive is required, a resistor  $R_{\text{out}}$  may be connected between the output and the negative supply. The minimum load which may be driven is:

$$R_{\text{load}} (\text{Kohm}) = 2.5 / (0.4 + V_{\text{ee}}/R_{\text{out}})$$

where  $R_{\text{out}}$  is in Kohms. The output is not short circuit protected. Therefore, if this pin is connected to outside of the equipment, a series resistor of  $470\text{ohms}$  in series with the output pin is needed.

## FINAL VCA

The final VCA is a low noise, low control voltage feedthrough design which does not require any trimming to null. Hence it is well suited for being controlled by fast transition envelopes without producing "pops" or "clicks".

The VCA signal input is a current summing input at a voltage of -2.1V, requiring an external series capacitor and resistor between the input signal voltage and input pin (Pin 13). The maximum input current should be limited to +-200uA. The value of input resistor is therefore:

$$R_{in} = V_{in}/400\mu A$$

The series capacitor is then chosen to give the desired -3dB low frequency corner with the selected resistor.

Somewhat lower distortion can be obtained with a lower maximum input current of +-50 to +-100uA at the expense of slightly lower signal-to-noise ratio and larger relative control feedthrough. Distortion also increases the lower input signal voltage; therefore the input signal voltage should be kept about 1Vpp.

The control scale is exponential from 0 to approx. 200mv, controlling the current gain from -100dB to about -20dB. Thereafter the current gain increases in a linear fashion until it reaches 0dB at +5V nominal. This slight rounded knee at the scale bottom allows an envelope to decay to zero with a natural exponential sound regardless of the small variations in VCA turn-on threshold.

As this VCA also has limited negative output voltage compliance (-1v max.), it is best to convert the output current to a voltage with a virtual ground summing op amp. Of course, if the output voltage needs to be no greater than 2V pp, the current-to-voltage conversion may be accomplished with a resistor to ground. The maximum voltage gain at +5V control is:

$$A_{max} = R_f/(R_{in} + 1.5K)$$

The outputs from several VCAs may be summed together by simply connecting the output pins together before converting to a voltage.





## C.6 Ensoniq DOC5503

Unfortunately, Ensoniq doesn't give away any information about the DOC chip so this information is derived from the Apple IIGS hardware manual.

### C.6.1 Common Registers

The DOC contains three registers which are common to all oscillators, in particular these are

#### Oscillator Interrupt Register (\$E0), OIR

This register contains the status of the DOC interrupt request (IRQ) pin and the number of the oscillator that generated the interrupt (if any). When an oscillator reaches the end of a wavetable and the enable interrupt bit for that oscillator has previously been set, the IRQ line and bit 7 of the OIR are then set, and the register number is entered in bits 1 through 5 of the OIR. Bits 6 and 0 are reserved and should not be modified.

#### Oscillator Enable Register (\$E1), OER

The OER controls the number of oscillators that are operating at a particular time. To enable one or more oscillators, multiply the desired quantity of oscillators (up to 32) by 2 and enter the number in this register. You may enter any number from 2 to 64, which will enable the corresponding oscillators in sequential order – low-numbered oscillators cannot be skipped in order to enable a higher-numbered one.

A minimum of one oscillator is always enabled, which is also the reset default.

#### A/D Converter Register (\$E2), ADCR

The ADCR contains the output value of the successive-approximation analog-to-digital converter. An analog-input signal with a maximum amplitude of  $2.5V_{ss}$  applied to pin can be sampled, the result of the conversion resides in the ADCR at the completion of the conversion. Reading this register initiates the 31-microsecond conversion process. If this register is read before the end of the conversion process, the value will be lost and a new conversion will begin.

### C.6.2 DOC registers for individual Oscillators

#### The Frequency High/Low registers (FRL: \$00-\$1F, FRH: \$20-\$3F)

The FRH and FRL registers are concatenated to create a 16-bit value for each oscillator. This frequency value determines the speed at which

the wavetable is read from memory. This speed indirectly determines the frequency of the output signal at the speaker.

The relationship between output signal frequency, wavetable scan rate, and the FRH/L register values is

$$\text{output frequency} = \frac{\text{scanrate}}{2^{(17+\text{wave address resolution})}} * F_{HL} \quad (\text{C.1})$$

where the scan rate is equal to  $\frac{894.7186\text{kHz}}{2^{\text{number of enabled oscillators}}}$ .

#### **The Volume registers (\$40-\$5F), VR**

The VR contain an oscillator's volume value. The current wavetable data byte is multiplied by the 8-bit volume value to obtain the oscillator final output level.

#### **The Oscillator Data registers (\$60-\$7F), ODR**

These registers contain an oscillator's last read value, i.e. the value of the most recently played byte. The ODR can be used e.g. for amplitude modulation.

#### **The Wavetable Pointer registers (\$80-\$9F), WPR**

This set of registers contains the beginning page number of the oscillator's wavetable. All wavetables must begin on a page boundary, i.e. the first byte of the page, and cannot wrap around to low memory addresses. The value in this register is used in the final address calculation.

#### **Oscillator Control Registers (\$A0-\$BF), OCR**

Each OCR controls all functions of each oscillator, including which of 16 optional external analog multiplexer channels an oscillator will use (the SQ80 uses only 8 of these channels), whether or not an oscillator may generate an interrupt and the oscillator's mode of operation.

The oscillator may function in one of several modes:

**Free Run Mode:** The oscillator starts at the beginning of the wavetable and repeats the same wavetable. It will halt when the halt bit is set or when a 0 is encountered in the table data.

**One Shot Mode:** The oscillator starts at the beginning of the wavetable, stepping through it only once, and stopping at the end of the table.

**Sync Mode:** You enable this mode by selecting even/odd pairs of oscillators (a lower even-numbered oscillator paired with an adjacent higher-numbered oscillator). When the even-numbered oscillator begins its wavetable, the odd-mate oscillator will synchronize and begin its wavetable simultaneously.

**Swap Mode:** Uses even/odd pairs of oscillators (same pairing as above). The enabled oscillator pair runs in One shot Mode. When it reaches the end of its wavetable, it resets its accumulator to 0, sets its halt bit, and clears the halt bit of its mate. This mode enables play-back of waves bigger than 32kB - unfortunately, it's quite unusable in the SQ80 since the DOC isn't allowed to generate interrupts. Thus the maximum wave size would be limited to 64kB.

Bit	Value	Description
7-4	-	Channel address bits, these determine to which demultiplexer output channel (provided by optional external demultiplexer hardware) this oscillator will be directed.
3	1	Interrupts enabled: An interrupt flag will be set in the DOC's OIR and the DOC will assert the IRQ signal when an oscillator generates an interrupt.
	0	Interrupts disabled
2-1	-	Oscillator mode: Select the mode desired by setting these two bits as follows: 00 Free Run 01 One Shot 10 Sync 11 Swap
0	1	Halted oscillator: This is a read/write bit. To halt the oscillator, set this bit. Certain modes (Sync, Swap) will halt the oscillator and set this bit automatically after completion.
	0	Running oscillator: This bit is cleared if the corresponding oscillator is currently enabled.

Table C.14: Bits in the Oscillator Control Register

### The Wavetable Size registers (\$C0-\$DF), WSR

The WSR control the size of the individual wavetable each oscillator will access. The size of the wavetable for each oscillator can vary from 256 bytes to 32kB in eight discrete steps as shown in table C.15.

Bit	Value	Description
7	-	reserved
6	-	Extended addressing, used for switching between sound banks of 64kB. In the SQ80 bit 7 of the OCR is used. Instead, this bit can be used for switching between sound banks of 128kB giving room for a possible waveform expansion unit.
5-3	-	Table size: The wavetables may extend up to 32kB in size but in discrete steps only shown in the following list: 000 256 bytes 001 512 bytes 010 1024 bytes 011 2048 bytes 100 4096 bytes 101 8192 bytes 110 16384 bytes 111 32768 bytes
2-0	-	Address Bus resolution: With this register you determine (corresponding to the wavetable size) which accumulator bits are used for wavetable addressing as shown in the following list: 000 A16 to A9-A2 001 A17 to A10-A3 010 A18 to A11-A4 011 A19 to A12-A5 100 A20 to A13-A6 101 A21 to A14-A7 110 A22 to A15-A8 111 A23 to A16-A9

Table C.15: Bits in the Wavetable Size register

In the SQ80 this register is used for octave shifting, for “neutral” playback set this bits to the same value as the table size.

### C.6.3 Wavetable Address Generation

The final wavetable address is generated according to figure C.3.

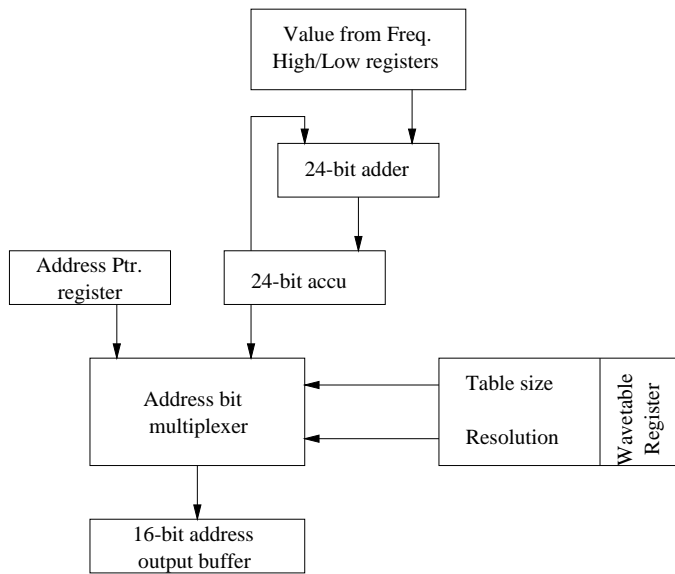


Figure C.3: Wavetable Address Generation

CLK	1	5503DOC	40	BS
/RAS	2		39	Vcc
/CAS	3		38	A0
Q	4		37	A1
E	5		36	A2
CSRB	6		35	A3
CA0	7		34	A4
CA1	8		33	A5
CA2	9		32	A6
CA3	10		31	A7
VVREF	11		30	A8/D0
VLFBK	12		29	A9/D1
VOL-	13		28	A10/D2
WVREF	14		27	A11/D3
Sig+	15		26	A12/D4
Sig-	16		25	A13/D5
A/D	17		24	A14/D6
/IRQ	18		23	A15/D7
GND	19		22	R/W
/RESET	20		21	/CS

Figure C.4: Pinout

### C.6.4 Pinout

Nothing unusual here. The A/D lines serve two purposes, one is communication with the host processor, the other is accessing waveform memory. As you can see it's a multiplexed A/D bus designed to directly access DRAM memory but it's also capable driving SRAM, for this you have to latch the

addresses using RAS# as ALE#.

Analog multiplexing is done by the CAx pins, the validity of the current pattern is shown by CSTRB. The SQ80 “abuses” CA3 for switching between wave ROM 0 and 1, normally this is done by BS.

**C.7 MC/HD 68B09E CPU**





# HD6809E, HD68A09E, HD68B09E MPU (Micro Processing Unit)

The HD6809E is a revolutionary high performance 8-bit microprocessor which supports modern programming techniques such as position independence, reentrancy, and modular programming.

This third-generation addition to the HMCS6800 family has major architectural improvements which include additional registers, instructions and addressing modes.

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6809E has the most complete set of addressing modes available on any 8-bit microprocessor today.

The HD6809E has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications. External clock inputs are provided to allow synchronization with peripherals, systems or other MPU's.

### HD6800 COMPATIBLE

- Hardware – Interfaces with All HMCS6800 Peripherals
- Software – Upward Source Code Compatible Instruction Set and Addressing Modes

### ARCHITECTURAL FEATURES

- Two 16-bit Index Registers
- Two 16-bit Indexable Stack Pointers
- Two 8-bit Accumulators can be Concatenated to Form One 16-Bit Accumulator
- Direct Page Register Allows Direct Addressing Throughout Memory

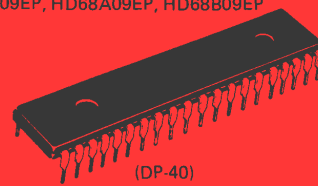
### HARDWARE FEATURES

- External Clock Inputs, E and Q, Allow Synchronization
- TSC Input Controls Internal Bus Buffers
- LIC Indicates Opcode Fetch
- AVMA Allows Efficient Use of Common Resources in A Multiprocessor System
- BUSY is a Status Line for Multiprocessing
- Fast Interrupt Request Input Stacks Only Condition Code Register and Program Counter
- Interrupt Acknowledge Output Allows Vectoring By Devices
- SYNC Acknowledge Output Allows for Synchronization to External Event
- Single Bus-Cycle RESET
- Single 5-Volt Supply Operation
- NMI Blocked After RESET Until After First Load of Stack Pointer
- Early Address Valid Allows Use With Slower Memories
- Early Write-Data for Dynamic Memories

### SOFTWARE FEATURES

- 10 Addressing Modes
  - HMCS6800 Upward Compatible Addressing Modes
  - Direct Addressing Anywhere in Memory Map
  - Long Relative Branches
  - Program Counter Relative
  - True Indirect Addressing
  - Expanded Indexed Addressing:
    - 0, 5, 8, or 16-bit Constant Offsets
    - 8, or 16-bit Accumulator Offsets

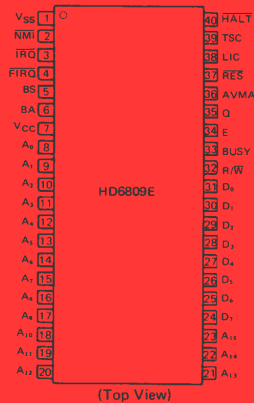
HD6809EP, HD68A09EP, HD68B09EP



Auto-Increment/Decrement by 1 or 2

- Improved Stack Manipulation
- 1464 Instruction with Unique Addressing Modes
- 8 x 8 Unsigned Multiply
- 16-bit Arithmetic
- Transfer/Exchange All Registers
- Push/Pull Any Registers or Any Set of Registers
- Load Effective Address

### PIN ARRANGEMENT



## HD6809E, HD68A09E, HD68B09E

## ■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V <sub>CC</sub> *	-0.3 ~ +7.0	V
Input Voltage	V <sub>IN</sub> *	-0.3 ~ +7.0	V
Operating Temperature Range	T <sub>opr</sub>	-20 ~ +75	°C
Storage Temperature Range	T <sub>stg</sub>	-55 ~ +150	°C

\* With respect to V<sub>SS</sub> (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

## ■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	unit	
Supply Voltage	V <sub>CC</sub> *	4.75	5.0	5.25	V	
Input Voltage	Logic, O, $\overline{\text{RES}}$	V <sub>IL</sub> *	-0.2	—	0.8	V
	E	V <sub>ILC</sub> *	-0.3	—	0.4	V
	Logic	V <sub>IH</sub> *	2.2	—	V <sub>CC</sub> *	V
	$\overline{\text{RES}}$		4.0	—	V <sub>CC</sub> *	V
	E	V <sub>IHC</sub> *	V <sub>CC</sub> * - 0.75	—	V <sub>CC</sub> * + 0.3	V
Operating Temperature	T <sub>opr</sub>	-20	25	75	°C	

\* With respect to V<sub>SS</sub> (SYSTEM GND)

## ■ ELECTRICAL CHARACTERISTICS

■ DC CHARACTERISTICS (V<sub>CC</sub> = 5.0V ±5%, V<sub>SS</sub> = 0V, T<sub>a</sub> = -20 ~ +75°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6809E			HD68A09E			HD68B09E			Unit		
			min	typ*	max	min	typ*	max	min	typ*	max			
Input "High" Voltage	Logic, O	V <sub>IH</sub>	2.2	—	V <sub>CC</sub>	2.2	—	V <sub>CC</sub>	2.2	—	V <sub>CC</sub>	V		
	$\overline{\text{RES}}$	V <sub>IHR</sub>	4.0	—	V <sub>CC</sub>	4.0	—	V <sub>CC</sub>	4.0	—	V <sub>CC</sub>	V		
	E	V <sub>IHC</sub>	V <sub>CC</sub> - 0.75	—	V <sub>CC</sub> + 0.3	V <sub>CC</sub> - 0.75	—	V <sub>CC</sub> + 0.3	V <sub>CC</sub> - 0.75	—	V <sub>CC</sub> + 0.3	V		
Input "Low" Voltage	Logic, O, $\overline{\text{RES}}$	V <sub>IL</sub>	-0.2	—	0.8	-0.2	—	0.8	-0.2	—	0.8	V		
	E	V <sub>ILC</sub>	-0.3	—	0.4	-0.3	—	0.4	-0.3	—	0.4	V		
Input Leakage Current	Logic, O, $\overline{\text{RES}}$	I <sub>in</sub>	V <sub>in</sub> = 0 ~ 5.25V, V <sub>CC</sub> = max		-2.5	—	2.5	-2.5	—	2.5	-2.5	—	2.5	μA
	E		-100	—	100	-100	—	100	-100	—	100	μA		
Output "High" Voltage	D <sub>0</sub> ~ D <sub>7</sub>	V <sub>OH</sub>	I <sub>Load</sub> = -205μA, V <sub>CC</sub> = min		2.4	—	—	2.4	—	—	2.4	—	—	V
	A <sub>0</sub> ~ A <sub>15</sub> , R/W		I <sub>Load</sub> = -145μA, V <sub>CC</sub> = min		2.4	—	—	2.4	—	—	2.4	—	—	V
	BA, BS, LIC, AVMA, BUSY		I <sub>Load</sub> = -100μA, V <sub>CC</sub> = min		2.4	—	—	2.4	—	—	2.4	—	—	V
Output "Low" Voltage	V <sub>OL</sub>	I <sub>Load</sub> = 2mA, V <sub>CC</sub> = min		—	—	0.5	—	—	0.5	—	—	0.5	V	
Power Dissipation	P <sub>D</sub>	—		—	—	1.0	—	—	1.0	—	—	1.0	W	
Input Capacitance	D <sub>0</sub> ~ D <sub>7</sub> , Logic Input, O, $\overline{\text{RES}}$	C <sub>in</sub>	V <sub>in</sub> = 0V, T <sub>a</sub> = 25°C, f = 1MHz		—	10	15	—	10	15	—	10	15	pF
	E		—	30	50	—	30	50	—	30	50	pF		
Output Capacitance	A <sub>0</sub> ~ A <sub>15</sub> , R/W, BA, BS, LIC, AVMA, BUSY	C <sub>out</sub>	V <sub>in</sub> = 0V, T <sub>a</sub> = 25°C, f = 1MHz		—	10	15	—	10	15	—	10	15	pF
Frequency of Operation	E, O	f	0.1	—	1.0	0.1	—	1.5	0.1	—	2.0	MHz		
Three-State (Off State) Input Current	D <sub>0</sub> ~ D <sub>7</sub>	I <sub>TSI</sub>	V <sub>in</sub> = 0.4 ~ 2.4V, V <sub>CC</sub> = max		-10	—	10	-10	—	10	-10	—	10	μA
	A <sub>0</sub> ~ A <sub>15</sub> , R/W		-100	—	100	-100	—	100	-100	—	100	μA		

\* T<sub>a</sub> = 25°C, V<sub>CC</sub> = 5V

HD6809E, HD68A09E, HD68B09E

● AC CHARACTERISTICS (V<sub>CC</sub> = 5.0V ±5%, V<sub>SS</sub> = 0V, T<sub>a</sub> = -20 ~ +75°C, unless otherwise noted.)  
**READ/WRITE TIMING**

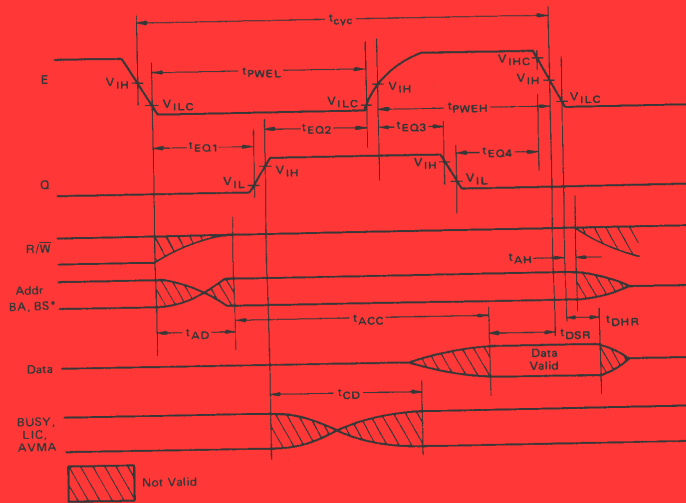
Item	Symbol	Test Condition	HD6809E			HD68A09E			HD68B09E			Unit	
			min	typ	max	min	typ	max	min	typ	max		
Cycle Time	t <sub>cyc</sub>	Fig. 1, 2, 7 ~ 10, 14 and 17	1000	—	10000	667	—	10000	500	—	10000	ns	
Peripheral Read Access Times t <sub>cyc</sub> - t <sub>EF</sub> - t <sub>AD</sub> - t <sub>DSR</sub> = t <sub>ACC</sub>	t <sub>ACC</sub>		695	—	—	440	—	—	330	—	—	ns	
Data Setup Time (Read)	t <sub>DSR</sub>		80	—	—	60	—	—	40	—	—	ns	
Input Data Hold Time	t <sub>DHR</sub>		10	—	—	10	—	—	10	—	—	ns	
Output Data Hold Time	T <sub>a</sub> = 0 ~ +75°C T <sub>a</sub> = -20 ~ 0°C		t <sub>DHW</sub>	30	—	—	30	—	—	30	—	—	ns
				20	—	—	20	—	—	20	—	—	ns
Address Hold Time (Address, R/W)	T <sub>a</sub> = 0 ~ +75°C T <sub>a</sub> = -20 ~ 0°C		t <sub>AH</sub>	20	—	—	20	—	—	20	—	—	ns
				10	—	—	10	—	—	10	—	—	ns
Address Delay	t <sub>AD</sub>		—	—	200	—	—	140	—	—	120	ns	
Data Delay Time (Write)	t <sub>DDW</sub>		—	—	200	—	—	140	—	—	110	ns	
E Clock "Low"	t <sub>PWEL</sub>		450	—	9500	295	—	9500	210	—	9500	ns	
E Clock "High" (Measured at V <sub>IH</sub> )	t <sub>PWEH</sub>		450	—	9500	280	—	9500	220	—	9500	ns	
E Rise and Fall Time	t <sub>Er</sub> , t <sub>Ef</sub>		—	—	25	—	—	25	—	—	20	ns	
Q Clock "High"	t <sub>PWQH</sub>		450	—	9500	280	—	9500	220	—	9500	ns	
Q Rise and Fall Time	t <sub>Qr</sub> , t <sub>Qf</sub>		—	—	25	—	—	25	—	—	20	ns	
E "Low" to Q Rising	t <sub>EQ1</sub>		200	—	—	130	—	—	100	—	—	ns	
Q "High" to E Rising	t <sub>EQ2</sub>		200	—	—	130	—	—	100	—	—	ns	
E "High" to Q Falling	t <sub>EQ3</sub>		200	—	—	130	—	—	100	—	—	ns	
Q "Low" to E Falling	t <sub>EQ4</sub>		200	—	—	130	—	—	100	—	—	ns	
Interrupts HALT, RES and TSC Setup Time	t <sub>PCT</sub>		200	—	—	140	—	—	110	—	—	ns	
TSC Drive to Valid Logic Levels	t <sub>TSA</sub>	—	—	210	—	—	150	—	—	120	ns		
TSC Release MOS Buffers to High Impedance	t <sub>TSR</sub>	—	—	200	—	—	140	—	—	110	ns		
TSC Three-State Delay	t <sub>TSD</sub>	—	—	120	—	—	85	—	—	80	ns		
Control Delay (BUSY, LIC)	t <sub>CD</sub>	—	—	300	—	—	250	—	—	200	ns		
Control Delay (AVMA*)	t <sub>CD</sub>	—	—	300	—	—	270	—	—	240	ns		
Processor Control Rise/Fall	t <sub>PCr</sub> , t <sub>PCf</sub>	—	—	100	—	—	100	—	—	100	ns		
TSC Input Delay	t <sub>PCT</sub>	10	—	—	10	—	—	10	—	—	ns		

\* AVMA drives a not-valid data before providing correct output, so spec t<sub>CD max</sub> = 270 nsec (HD68A09E) and 240 nsec (HD68B09E) are applied to this signal. When this delay time causes a problem in user's application, please use D-type latch to get stable output.

HITACHI

287

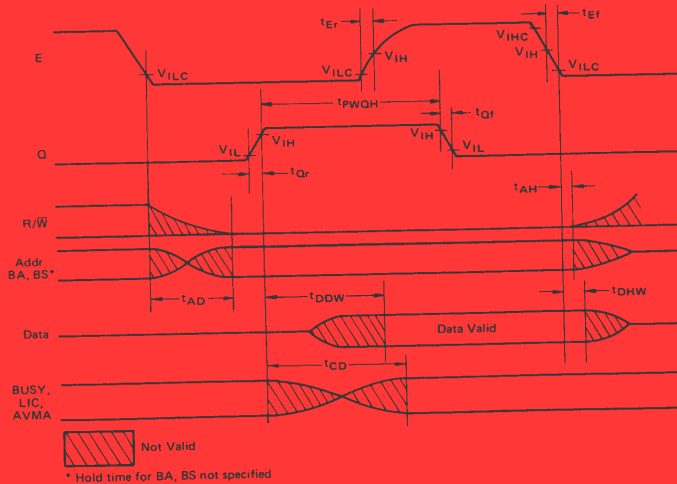
HD6809E, HD68A09E, HD68B09E



\* Hold time for BA, BS not specified

(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" =  $V_{IHmin}$  and logic "Low" =  $V_{ILmax}$  unless otherwise specified.

Figure 1 Read Data from Memory or Peripherals



\* Hold time for BA, BS not specified

(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" =  $V_{IHmin}$  and logic "Low" =  $V_{ILmax}$  unless otherwise specified.

Figure 2 Write Data to Memory or Peripherals

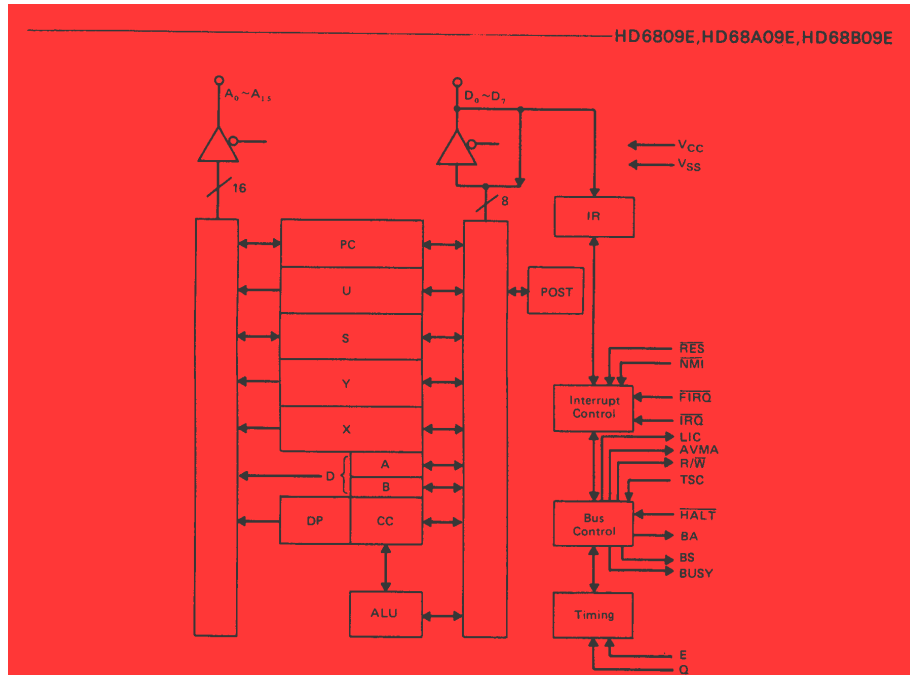
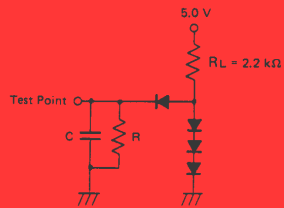


Figure 3 HD6809E Expanded Block Diagram



C = 30 pF for BA, BS, LIC, AVMA, BUSY  
 130 pF for D<sub>0</sub> ~ D<sub>7</sub>  
 90 pF for A<sub>0</sub> ~ A<sub>15</sub>, R/W  
 R = 11.7 kΩ for D<sub>0</sub> ~ D<sub>7</sub>  
 16.5 kΩ for A<sub>0</sub> ~ A<sub>15</sub>, R/W  
 24 kΩ for BA, BS, LIC, AVMA, BUSY

All diodes are 1S2074(Ⓜ) or equivalent.  
 C includes stray capacitance.

Figure 4 Bus Timing Test Load

■ PROGRAMMING MODEL

As shown in Figure 5, the HD6809E adds three registers to the set available in the HD6800. The added registers include a Direct Page Register, the User Stack pointer and a second Index Register.

● Accumulators (A, B, D)

The A and B registers are general purpose accumulators which are used for arithmetic calculations and manipulation of data.

Certain instructions concatenate the A and B registers to form a single 16-bit accumulator. This is referred to as the D Register, and is formed with the A Register as the most significant byte.

● Direct Page Register (DP)

The Direct Page Register of the HD6809E serves to enhance the Direct Addressing Mode. The content of this register appears at the higher address outputs (A<sub>8</sub> ~ A<sub>15</sub>) during direct addressing instruction execution. This allows the direct mode to be used at any place in memory, under program control. To ensure HD6800 compatibility, all bits of this register are cleared during Processor Reset.

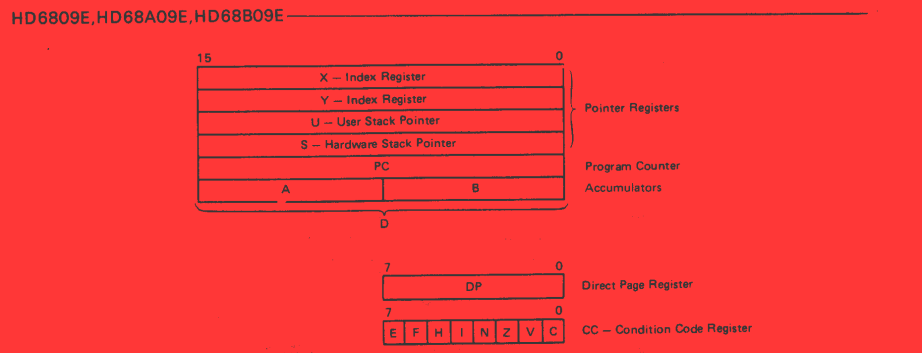


Figure 5 Programming Model of The Microprocessing Unit

- **Index Registers (X, Y)**

The Index Registers are used in indexed mode of addressing. The 16-bit address in this register takes part in the calculation of effective addresses. This address may be used to point to data directly or may be modified by an optional constant or register offset. During some indexed modes, the contents of the index register are incremented or decremented to point to the next item of tabular type data. All four pointer registers (X, Y, U, S) may be used as index registers.

- **Stack Pointer (U, S)**

The Hardware Stack Pointer (S) is used automatically by the processor during subroutine calls and interrupts. The User Stack Pointer (U) is controlled exclusively by the programmer thus allowing arguments to be passed to and from subroutines with ease. The U-register is frequently used as a stack marker. Both Stack Pointers have the same indexed mode addressing capabilities as the X and Y registers, but also support Push and Pull instructions. This allows the HD6809E to be used efficiently as a stack processor, greatly enhancing its ability to support higher level languages and modular programming.

(NOTE) The stack pointers of the HD6809E point to the top of the stack, in contrast to the HD6800 stack pointer, which pointed to the next free location on stack.

- **Program Counter (PC)**

The Program Counter is used by the processor to point to the address of the next instruction to be executed by the processor. Relative Addressing is provided allowing the Program Counter to be used like an index register in some situations.

- **Condition Code Register (CC)**

The Condition Code Register defines the state of the processor at any given time. See Figure 6.

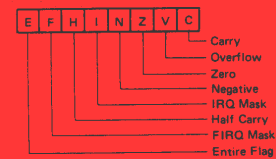


Figure 6 Condition Code Register Format

- **CONDITION CODE REGISTER DESCRIPTION**

- **Bit 0 (C)**

Bit 0 is the carry flag, and is usually the carry from the binary ALU. C is also used to represent a 'borrow' from subtract like instructions (CMP, NEG, SUB, SBC) and is the complement of the carry from the binary ALU.

- **Bit 1 (V)**

Bit 1 is the overflow flag, and is set to a one by an operation which causes a signed two's complement arithmetic overflow. This overflow is detected in an operation in which the carry from the MSB in the ALU does not match the carry from the MSB-1.

- **Bit 2 (Z)**

Bit 2 is the zero flag, and is set to a one if the result of the previous operation was identically zero.

- **Bit 3 (N)**

Bit 3 is the negative flag, which contains exactly the value of the MSB of the result of the preceding operation. Thus, a negative two's-complement result will leave N set to a one.

- **Bit 4 (I)**

Bit 4 is the  $\overline{\text{IRQ}}$  mask bit. The processor will not recognize interrupts from the  $\overline{\text{IRQ}}$  line if this bit is set to a one.  $\overline{\text{NMI}}$ ,  $\overline{\text{FIRQ}}$ ,  $\overline{\text{IRQ}}$ ,  $\overline{\text{RES}}$  and SW1 all set I to a one; SW2 and SW3 do not affect I.

- **Bit 5 (H)**

Bit 5 is the half-carry bit, and is used to indicate a carry from bit 3 in the ALU as a result of an 8-bit addition only (ADC or ADD). This bit is used by the DAA instruction to perform a BCD decimal add adjust operation. The state of this flag is undefined in all subtract-like instructions.

- **Bit 6 (F)**

Bit 6 is the  $\overline{\text{FIRQ}}$  mask bit. The processor will not recognize interrupts from the  $\overline{\text{FIRQ}}$  line if this bit is a one.  $\overline{\text{NMI}}$ ,  $\overline{\text{FIRQ}}$ , SW1, and  $\overline{\text{RES}}$  all set F to a one.  $\overline{\text{IRQ}}$ , SW2 and SW3 do not affect F.

- **Bit 7 (E)**

Bit 7 is the entire flag, and when set to a one indicates that the complete machine state (all the registers) was stacked, as opposed to the subset state (PC and CC). The E bit of the stacked CC is used on a return from interrupt (RTI) to determine the extent of the unstacking. Therefore, the current E left in the Condition Code Register represents past action.

- **HD6809E MPU SIGNAL DESCRIPTION**

- **Power (Vss, Vcc)**

Two pins are used to supply power to the part: Vss is ground or 0 volts, while Vcc is +5.0 V  $\pm 5\%$ .

- **Address Bus (A<sub>0</sub> ~ A<sub>15</sub>)**

Sixteen pins are used to output address information from the MPU onto the Address Bus. When the processor does not require the bus for a data transfer, it will output address  $\text{FFFF}_{16}$ ,  $\text{R}/\overline{\text{W}}$  = "High", and BS = "Low"; this is a "dummy access" or  $\overline{\text{VMA}}$  cycle. All address bus drivers are made high-impedance when output Bus Available (BA) is "High" or when TSC is asserted. Each pin will drive one Schottky TTL load or four LS TTL loads, and 90 pF. Refer to Figures 1 and 2.

- **Data Bus (D<sub>0</sub> ~ D<sub>7</sub>)**

These eight pins provide communication with the system bi-directional data bus. Each pin will drive one Schottky TTL load or four LS TTL loads, and 130 pF.

- **Read/Write (R/ $\overline{\text{W}}$ )**

This signal indicates the direction of data transfer on the data bus. A "Low" indicates that the MPU is writing data-onto the data bus.  $\text{R}/\overline{\text{W}}$  is made high impedance when BA is "High" or when TSC is asserted. Refer to Figures 1 and 2.

- **$\overline{\text{RES}}$**

A "Low" level on this Schmitt-trigger input for greater than one bus cycle will reset the MPU, as shown in Figure 7. The Reset vectors are fetched from locations  $\text{FFFE}_{16}$  and  $\text{FFFF}_{16}$  (Table 1) when Interrupt Acknowledge is true, ( $\overline{\text{BA}} \cdot \text{BS} = 1$ ). During initial power-on, the Reset line should be held "Low" until the clock input signals are fully operational.

Because the HD6809E Reset pin has a Schmitt-trigger input with a threshold voltage higher than that of standard peripherals, a simple R/C network may be used to reset the entire system.

This higher threshold voltage ensures that all peripherals are out of the reset state before the Processor.

Table 1 Memory Map for Interrupt Vectors

Memory Map for Vector Locations		Interrupt Vector Description
MS	LS	
FFFE	FFFF	$\overline{\text{RES}}$
FFFC	FFFD	$\overline{\text{NMI}}$
FFFA	FFFB	SW1
FFF8	FFF9	$\overline{\text{IRQ}}$
FFF6	FFF7	$\overline{\text{FIRQ}}$
FFF4	FFF5	SW2
FFF2	FFF3	SW3
FFF0	FFF1	Reserved

- **$\overline{\text{HALT}}$**

A "Low" level on this input pin will cause the MPU to stop running at the end of the present instruction and remain halted indefinitely without loss of data. When halted, the BA output is driven "High" indicating the buses are high impedance. BS is also "High" which indicates the processor is in the Halt state. While halted, the MPU will not respond to external real-time requests ( $\overline{\text{FIRQ}}$ ,  $\overline{\text{IRQ}}$ ) although  $\overline{\text{NMI}}$  or  $\overline{\text{RES}}$  will be latched for later response. During the Halt state Q and E should continue to run normally. A halted state ( $\text{BA} \cdot \text{BS} = 1$ ) can be achieved by pulling  $\overline{\text{HALT}}$  "Low" while  $\overline{\text{RES}}$  is still "Low". See Figure 8.

- **Bus Available, Bus Status (BA, BS)**

The Bus Available output is an indication of an internal control signal which makes the MOS buses of the MPU high impedance. When BA goes "Low", a dead cycle will elapse before the MPU acquires the bus. BA will not be asserted when TSC is active, thus allowing dead cycle consistency.

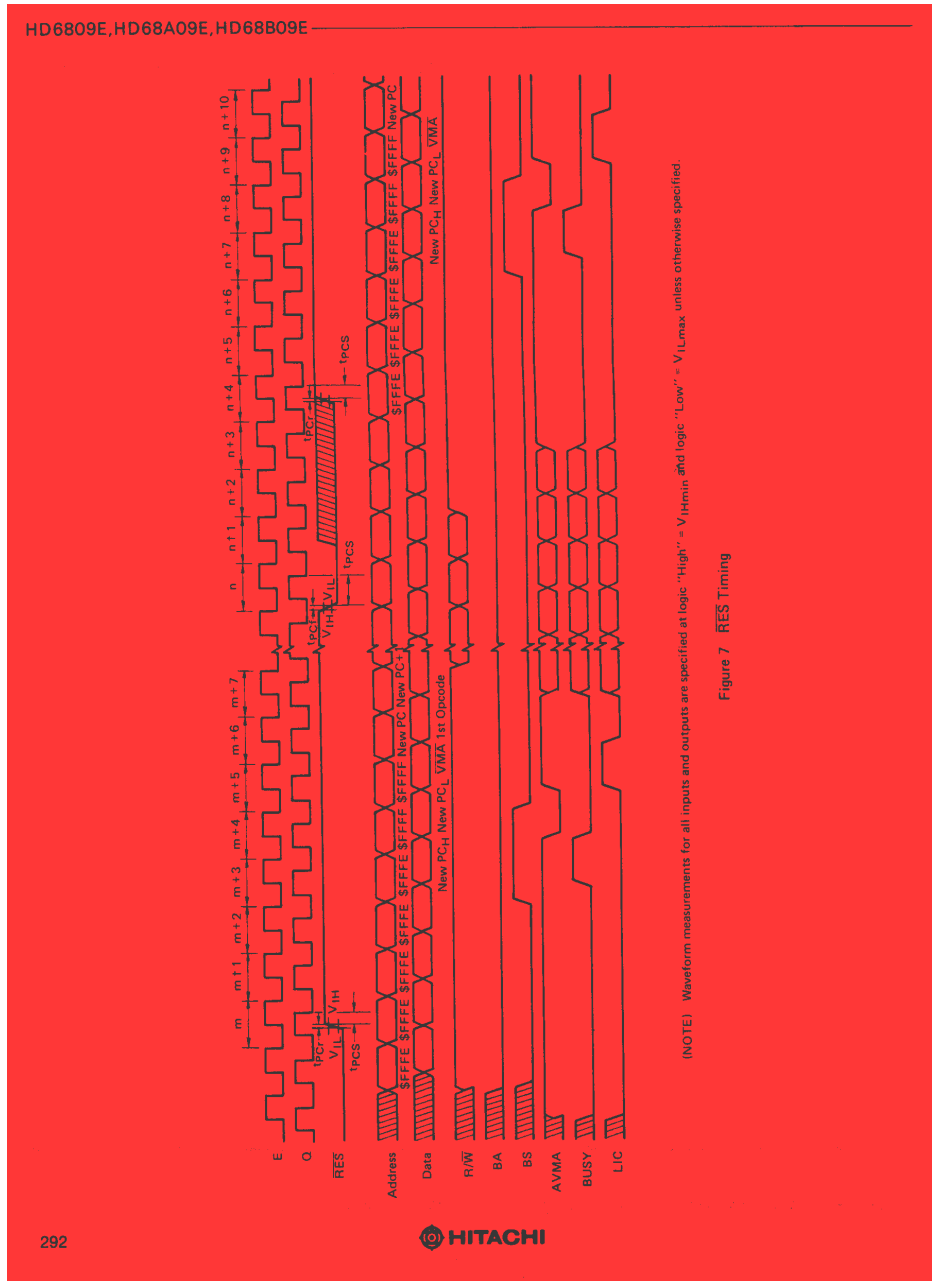
The Bus Status output signal, when decoded with BA, represents the MPU state (valid with leading edge of Q).

MPU State		MPU State Definition
BA	BS	
0	0	Normal (Running)
0	1	Interrupt or RESET Acknowledge
1	0	SYNC Acknowledge
1	1	HALT Acknowledge

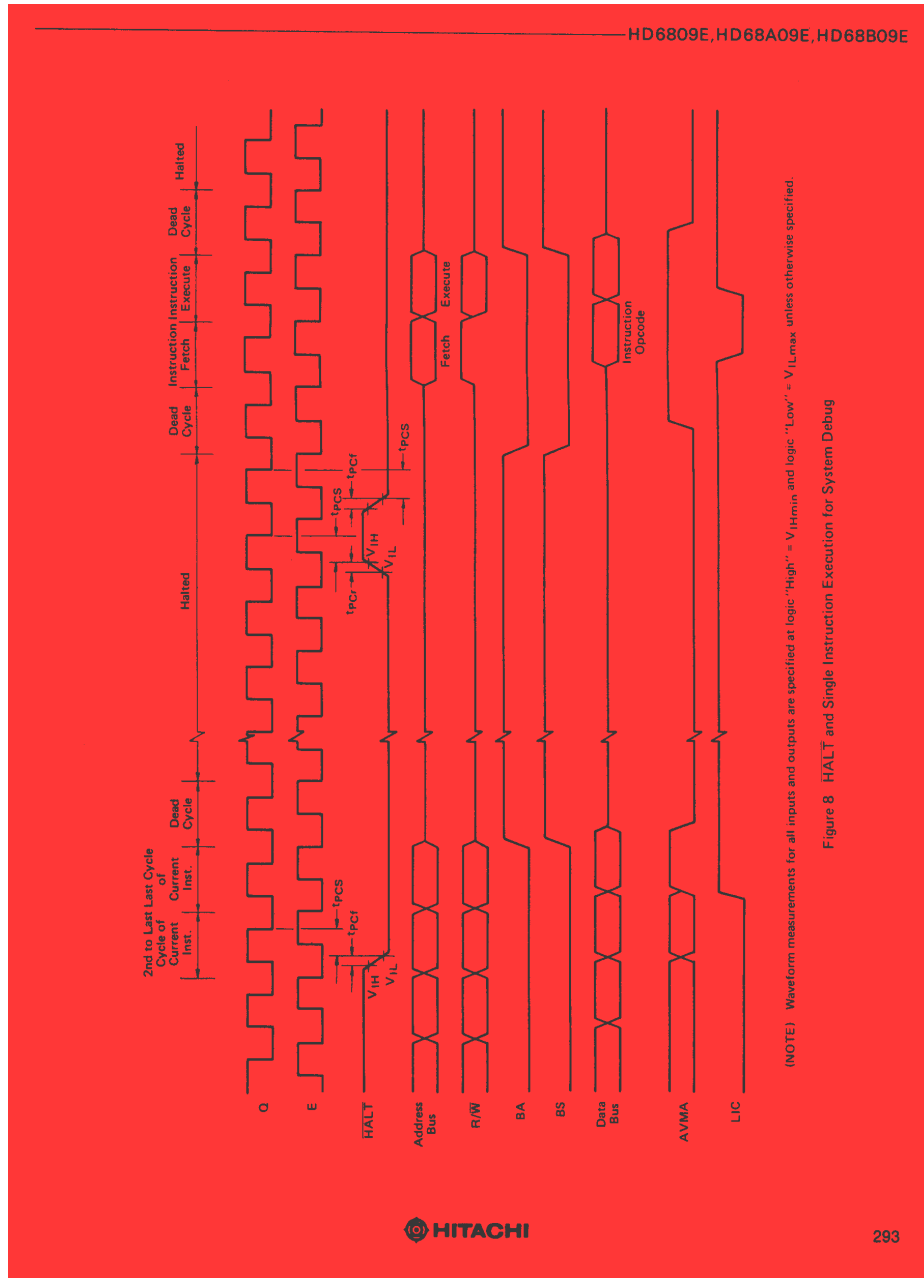
**Interrupt Acknowledge** is indicated during both cycles of a hardware-vector-fetch ( $\overline{\text{RES}}$ ,  $\overline{\text{NMI}}$ ,  $\overline{\text{FIRQ}}$ ,  $\overline{\text{IRQ}}$ , SW1, SW2, SW3). This signal, plus decoding of the lower four address lines, can provide the user with an indication of which interrupt level is being serviced and allow vectoring by device. See Table 1.

**Sync Acknowledge** is indicated while the MPU is waiting for external synchronization on an interrupt line.

**Halt Acknowledge** is indicated when the HD6809E is in a Halt condition.







## HD6809E, HD68A09E, HD68B09E

- **Non Maskable Interrupt ( $\overline{\text{NMI}}$ )\***

A negative transition on this input requests that a non-maskable interrupt sequence be generated. A non-maskable interrupt cannot be inhibited by the program, and also has a higher priority than  $\overline{\text{FIRQ}}$ ,  $\overline{\text{IRQ}}$  or software interrupts. During recognition of an  $\overline{\text{NMI}}$ , the entire machine state is saved on the hardware stack. After reset, an  $\overline{\text{NMI}}$  will not be recognized until the first program load of the Hardware Stack Pointer (S). The pulse width of  $\overline{\text{NMI}}$  low must be at least one E cycle. If the  $\overline{\text{NMI}}$  input does not meet the minimum set up with respect to Q, the interrupt will not be recognized until the next cycle. See Figure 9.

- **Fast-Interrupt Request ( $\overline{\text{FIRQ}}$ )\***

A "Low" level on this input pin will initiate a fast interrupt sequence, provided its mask bit (F) in the CC is clear. This sequence has priority over the standard Interrupt Request ( $\overline{\text{IRQ}}$ ), and is fast in the sense that it stacks only the contents of the condition code register and the program counter. The interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 10.

- **Interrupt Request ( $\overline{\text{IRQ}}$ )\***

A "Low" level input on this pin will initiate an Interrupt Request sequence provided the mask bit (I) in the CC is clear. Since  $\overline{\text{IRQ}}$  stacks the entire machine state it provides a slower response to interrupts than  $\overline{\text{FIRQ}}$ .  $\overline{\text{IRQ}}$  also has a lower priority than  $\overline{\text{FIRQ}}$ . Again, the interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 9.

\*  $\overline{\text{NMI}}$ ,  $\overline{\text{FIRQ}}$ , and  $\overline{\text{IRQ}}$  requests are sampled on the falling edge of Q. One cycle is required for synchronization before these interrupts are recognized. The pending interrupt(s) will not be serviced until completion of the current instruction unless a SYNC or CWAI condition is present. If  $\overline{\text{IRQ}}$  and  $\overline{\text{FIRQ}}$  do not remain "Low" until completion of the current instruction they may not be recognized. However,  $\overline{\text{NMI}}$  is latched and need only remain "Low" for one cycle.

- **Clock Inputs E, Q**

E and Q are the clock signals required by the HD6809E. Q must lead E; that is, a transition on Q must be followed by a similar transition on E after a minimum delay. Addresses will be valid from the MPU,  $t_{AD}$  after the falling edge of E, and data will be latched from the bus by the falling edge of E. While the Q input is fully TTL compatible, the E input directly drives internal MOS circuitry and, thus, requires levels above normal TTL levels. This approach minimizes clock skew inherent with an internal buffer. Timing and waveforms for E and Q are shown in Figures 1 and 2 while Figure 11 shows a simple clock generator for the HD6809E.

- **BUSY**

Busy will be "High" for the read and modify cycles of a read-modify-write instruction and during the access of the first byte

of a double-byte operation (e.g., LDX, STD, ADDD). Busy is also "High" during the first byte of any indirect or other vector fetch (e.g., jump extended, SWI indirect etc.).

In a multi-processor system, busy indicates the need to defer the arbitration of the next bus cycle to insure the integrity of the above operations. This difference provides the indivisible memory access required for a "test-and-set" primitive, using any one of several read-modify-write instructions.

Busy does not become active during PSH or PUL operations. A typical read-modify-write instruction (ASL) is shown in Figure 12. Timing information is given in Figure 13. Busy is valid  $t_{CD}$  after the rising edge of Q.

- **AVMA**

AVMA is the Advanced VMA signal and indicates that the MPU will use the bus in the following bus cycle. The predictive nature of the AVMA signal allows efficient shared-bus multi-processor systems. AVMA is "Low" when the MPU is in either a HALT or SYNC state. AVMA is valid  $t_{CD}$  after the rising edge of Q.

- **LIC**

LIC (Last Instruction Cycle) is "High" during the last cycle of every instruction, and its transition from "High" to "Low" will indicate that the first byte of an opcode will be latched at the end of the present bus cycle. LIC will be "High" when the MPU is Halted at the end of an instruction, (i.e., not in CWAI or RESET) in SYNC state or while stacking during interrupts. LIC is valid  $t_{CD}$  after the rising edge of Q.

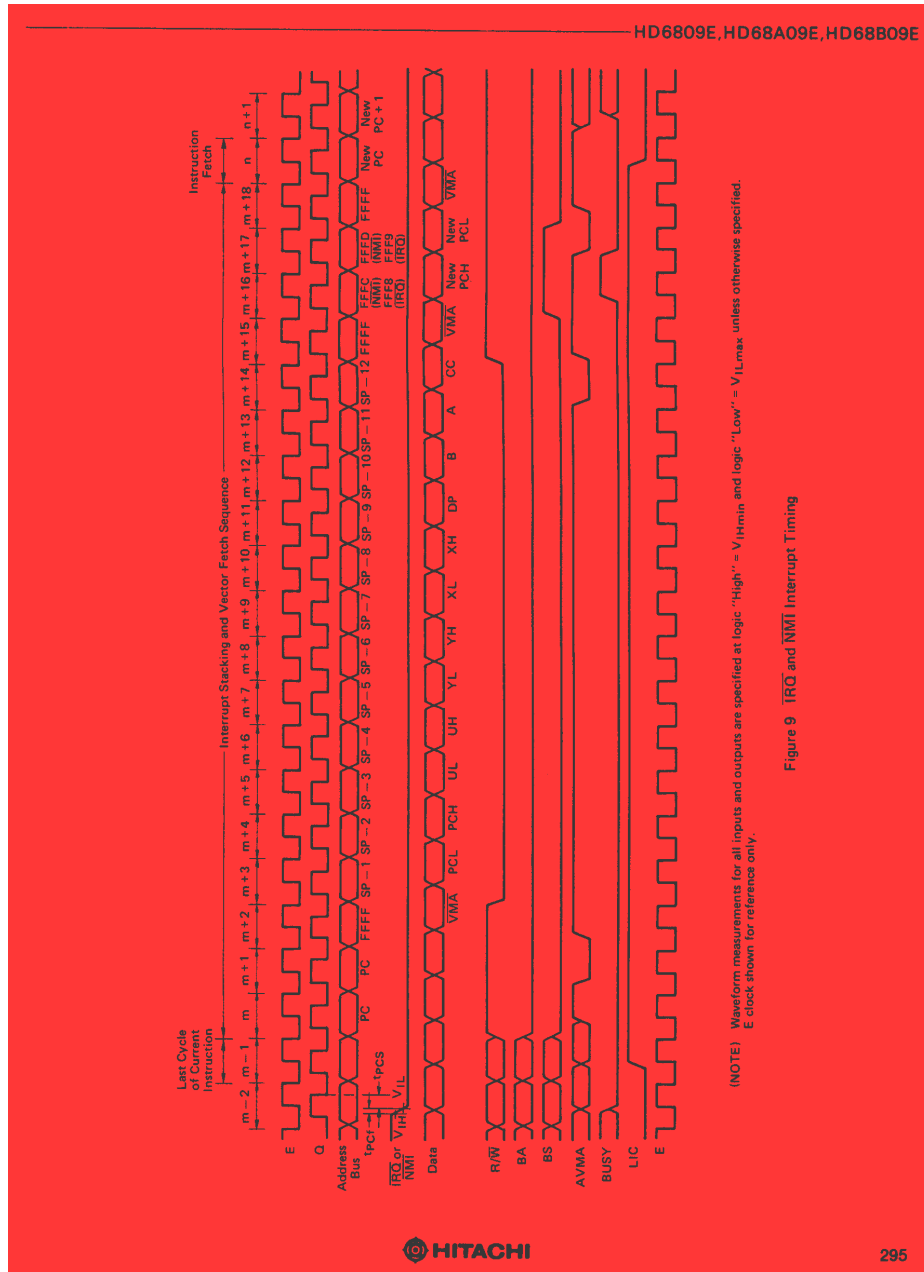
- **TSC**

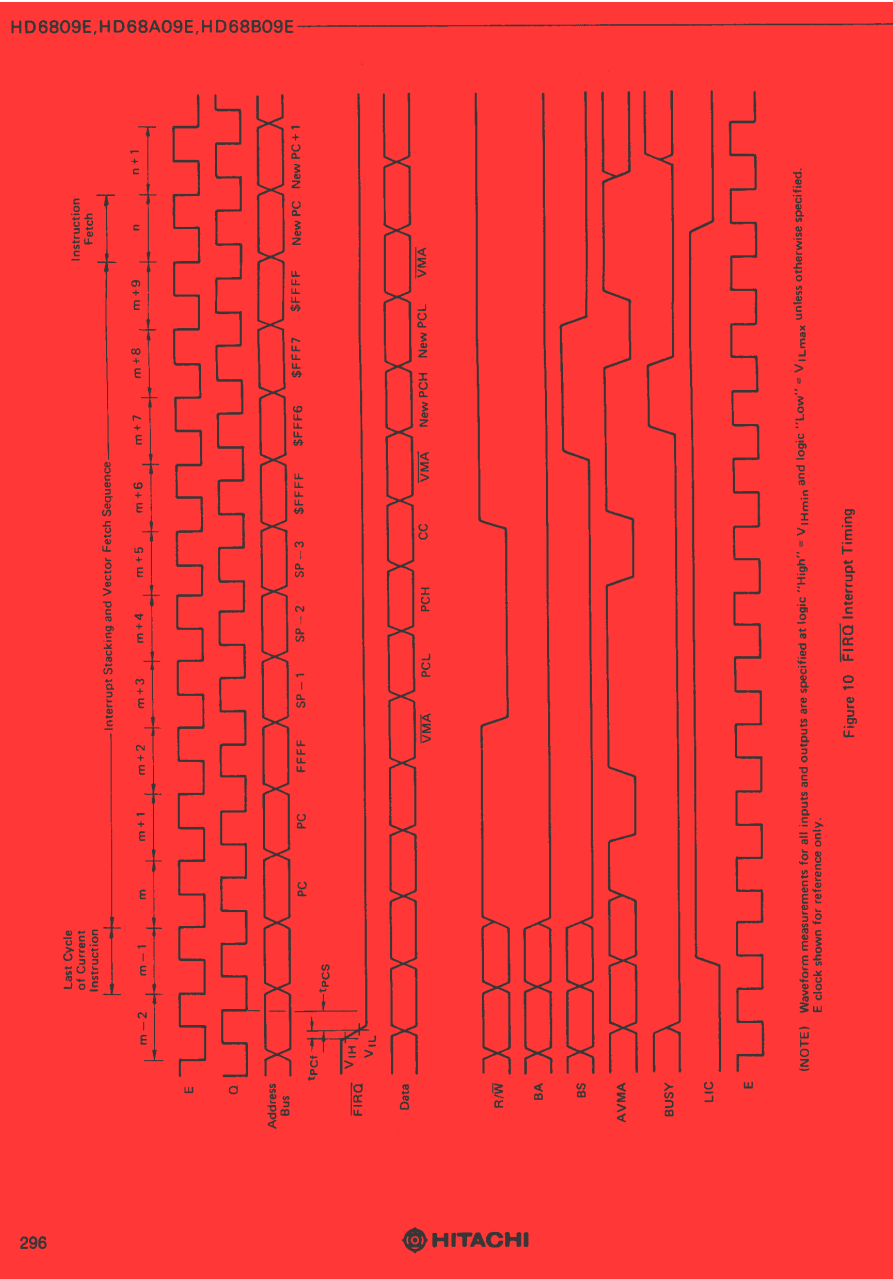
TSC (Three-State Control) will cause MOS address, data, and R/W buffers to assume a high-impedance state. The control signals (BA, BS, BUSY, AVMA and LIC) will not go to the high-impedance state. TSC is intended to allow a single bus to be shared with other bus masters (processors or DMA controllers).

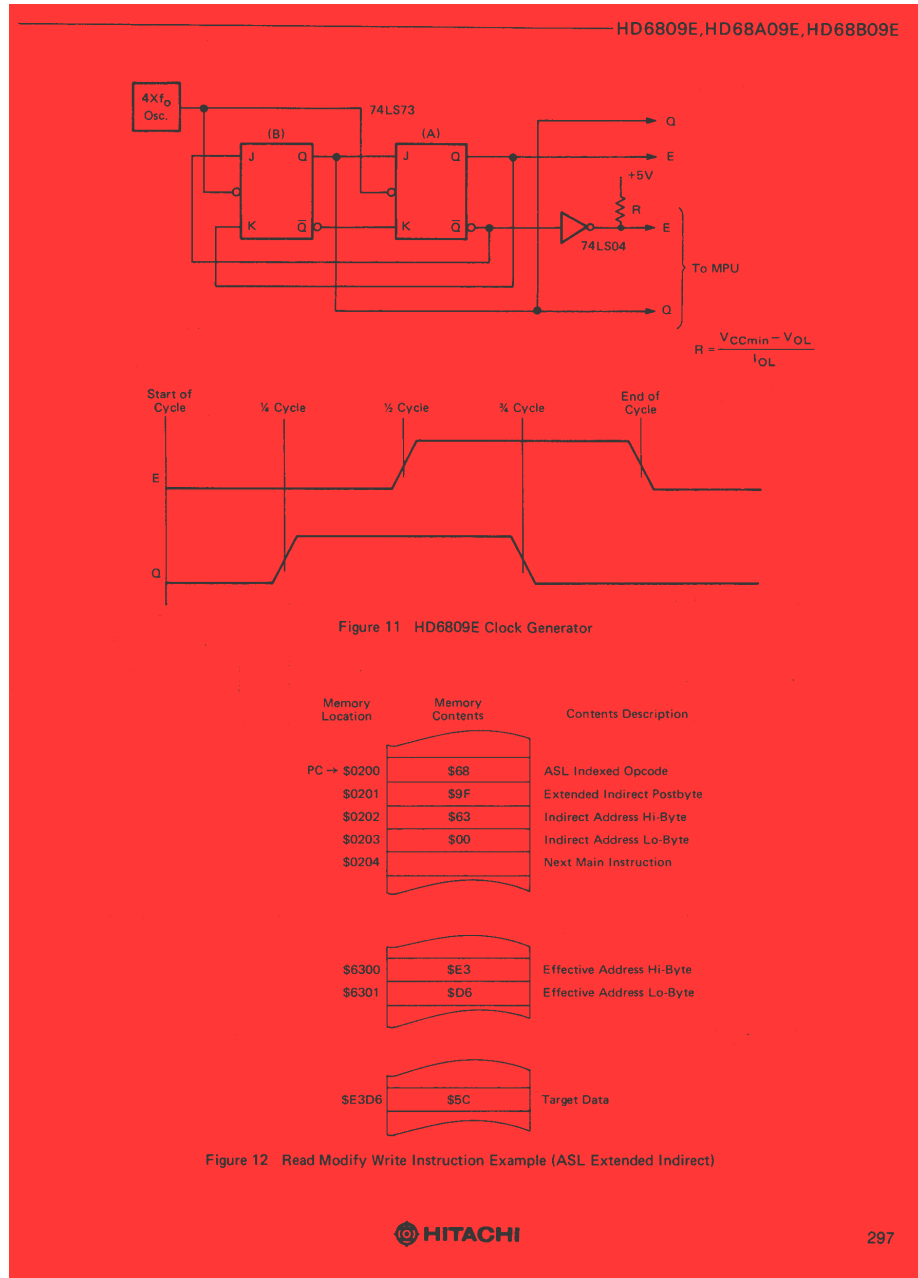
While E is "Low", TSC controls the address buffers and  $\overline{\text{R/W}}$  directly. The data bus buffers during a write operation are in a high-impedance state until Q rises at which time, if TSC is true, they will remain in a high-impedance state. If TSC is held beyond the rising edge of E, then it will be internally latched, keeping the bus drivers in a high-impedance state for the remainder of the bus cycle. See Figure 14.

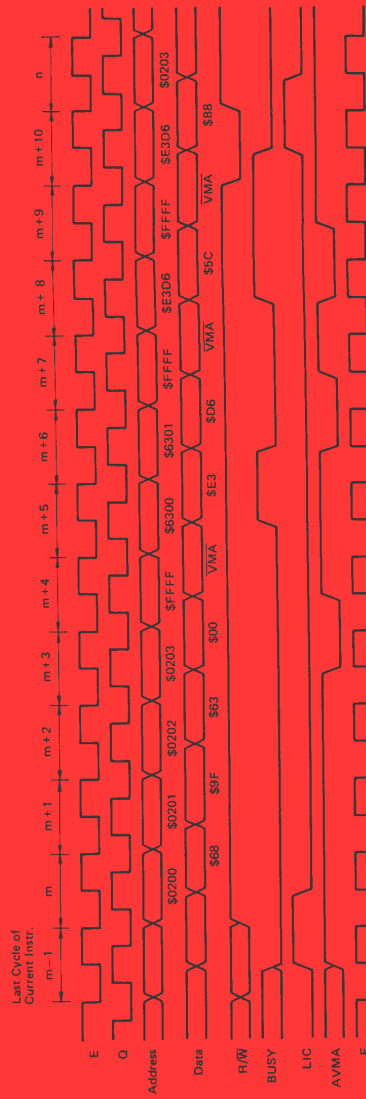
- **MPU Operation**

During normal operation, the MPU fetches an instruction from memory and then executes the requested function. This sequence begins after  $\overline{\text{RES}}$  and is repeated indefinitely unless altered by a special instruction or hardware occurrence. Software instructions that alter normal MPU operation are: SWI, SWI2, SWI3, CWAI, RTI and SYNC. An interrupt or HALT input can also alter the normal execution of instructions. Figure 15 illustrates the flow chart for the HD6809E.



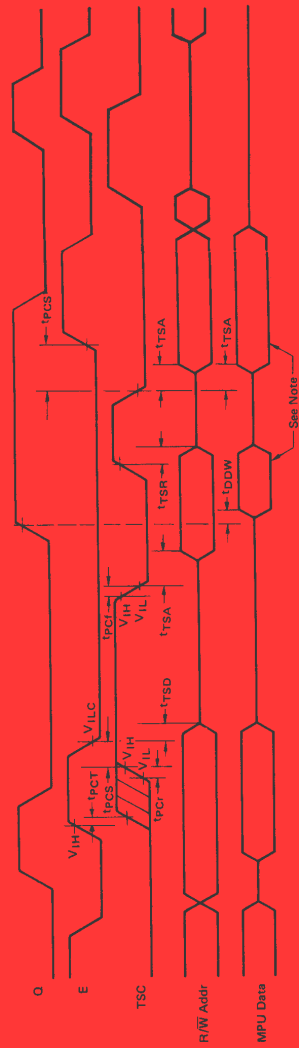






(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" =  $V_{IHmin}$  and logic "Low" =  $V_{LLmax}$  unless otherwise specified.

Figure 13 BUSY Timing (ASL Extended Indirect Instruction)



(NOTES) Data will be asserted by the MPU only during the interval while R/W is "Low" and E or O is "High". Waveform measurements for all inputs and outputs are specified at logic "High" =  $V_{IHmin}$  and logic "Low" =  $V_{LLmax}$  unless otherwise specified.

Figure 14 TSC Timing

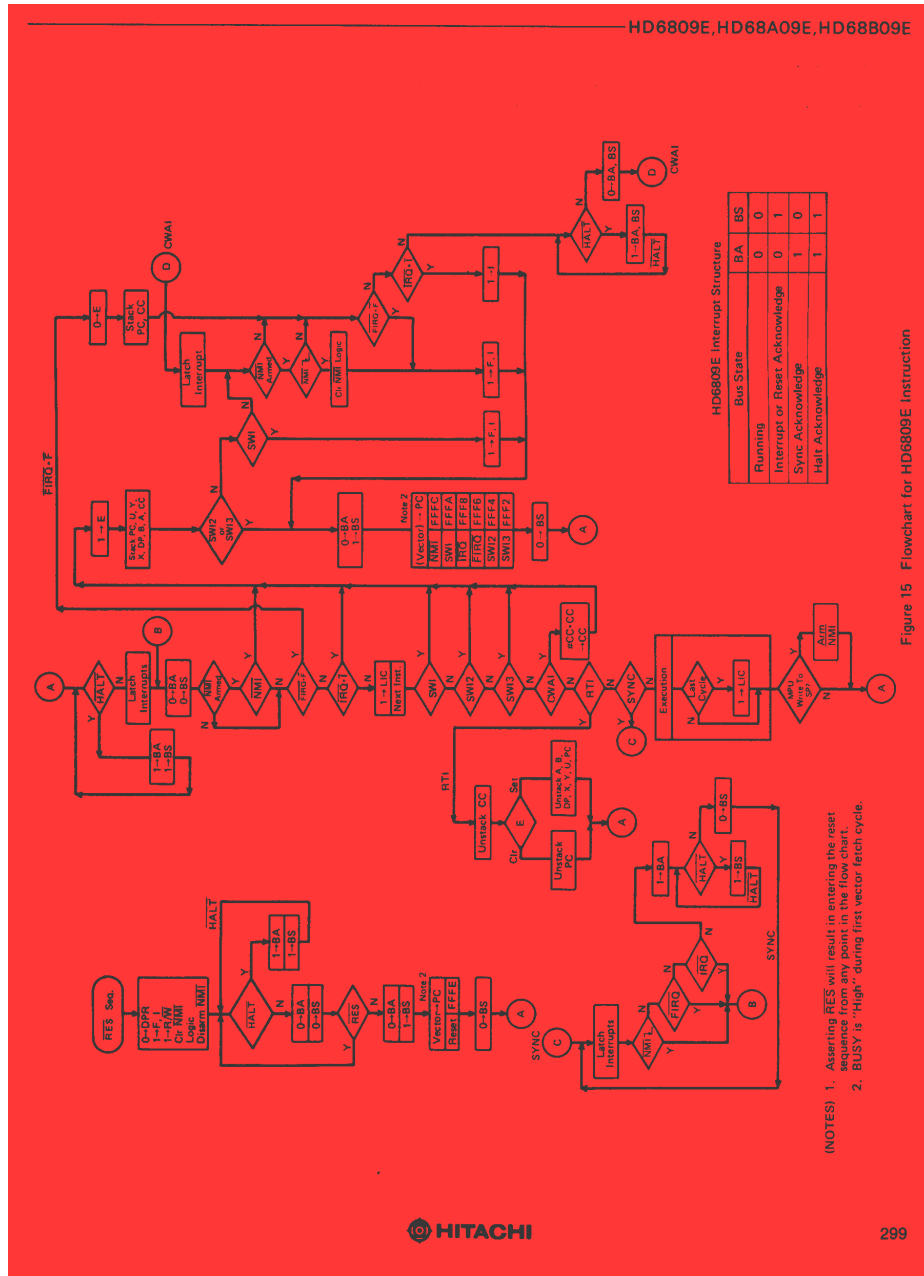


Figure 15 Flowchart for HD6809E Instruction

HD6809E, HD68A09E, HD68B09E

■ **ADDRESSING MODES**  
 The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6809E has the most complete set of addressing modes available on any microcomputer today. For example, the HD6809E has 59 basic instructions; however, it recognizes 1464 different variations of instructions and addressing modes. The addressing modes support modern programming techniques. The following addressing modes are available on the HD6809E:

- (1) Implied (Includes Accumulator)
- (2) Immediate
- (3) Extended
- (4) Extended Indirect
- (5) Direct
  - Zero-Offset
  - Constant Offset
  - Accumulator Offset
  - Auto Increment/Decrement
- (8) Indexed Indirect
- (9) Relative
- (10) Program Counter Relative

● **Implied (Includes Accumulator)**  
 In this addressing mode, the opcode of the instruction contains all the address information necessary. Examples of Implied Addressing are: ABX, DAA, SWI, ASRA, and CLR.B.

● **Immediate Addressing**  
 In Immediate Addressing, the effective address of the data is the location immediately following the opcode (i.e., the data to be used in the instruction immediately follows the opcode of the instruction). The HD6809E uses both 8 and 16-bit immediate values depending on the size of argument specified by the opcode. Examples of instructions with immediate Addressing are:

```
LDA #S20
LDX #SF000
LDY #CAT
```

(NOTE) # signifies immediate addressing, \$ signifies hexadecimal value.

● **Extended Addressing**  
 In Extended Addressing, the contents of the two bytes immediately following the opcode fully specify the 16-bit effective address used by the instruction. Note that the address generated by an extended instruction defines an absolute address and is not position independent. Examples of Extended Addressing include:

```
LDA CAT
STX MOUSE
LDD $2000
```

● **Extended Indirect**  
 As a special case of indexed addressing (discussed below), one level of indirection may be added to Extended Addressing. In Extended Indirect, the two bytes following the postbyte of an Indexed instruction contain the address of the data.

```
LDA [CAT]
LDX [SFFFE]
STU [DOG]
```

● **Direct Addressing**  
 Direct addressing is similar to extended addressing except that only one byte of address follows the opcode. This byte specifies the lower 8 bits of the address to be used. The upper 8 bits of the address are supplied by the direct page register. Since only one byte of address is required in direct addressing, this mode requires less memory and executes faster than extended addressing. Of course, only 256 locations (one page) can be accessed without redefining the contents of the DP register. Since the DP register is set to \$00 on Reset, direct addressing on the HD6809E is compatible with direct addressing on the HD6800. Indirection is not allowed in direct addressing. Some examples of direct addressing are:

```
LDA $30
SETDP $10 (Assembler directive)
LDB $1030
LDD <CAT
```

(NOTE) < is an assembler directive which forces direct addressing.

● **Register Addressing**  
 Some opcodes are followed by a byte that defines a register or set of registers to be used by the instruction. This is called a postbyte. Some examples of register addressing are:

```
TFR X, Y Transfer X into Y
EXG A, B Exchanges A with B
PSHS A, B, X, Y Push Y, X, B and A onto S
PULU X, Y, D Pull D, X, and Y from U
```

● **Indexed Addressing**  
 In all indexed addressing, one of the pointer registers (X, Y, U, S, and sometimes PC) is used in a calculation of the effective address of the operand to be used by the instruction. Five basic types of indexing are available and are discussed below. The postbyte of an indexed instruction specifies the basic type and variation of the addressing mode as well as the pointer register to be used. Figure 16 lists the legal formats for the postbyte. Table 2 gives the assembler form and the number of cycles and bytes added to the basic values for indexed addressing for each variation.

Post-Byte Register Bit								Indexed Addressing Mode
7	6	5	4	3	2	1	0	
0	R	R	d	d	d	d	d	EA = R + 5 Bit Offset
1	R	R	0	0	0	0	0	R =
1	R	R	1	0	0	0	1	R +
1	R	R	0	0	0	1	0	R -
1	R	R	1	0	0	1	1	R
1	R	R	1	0	1	0	0	EA = R + 0 Offset
1	R	R	1	0	1	0	1	EA = R + ACDB Offset
1	R	R	1	0	1	1	0	EA = R + ACCA Offset
1	R	R	1	1	0	0	0	EA = R + 8 Bit Offset
1	R	R	1	1	0	0	1	EA = R + 16 Bit Offset
1	R	R	1	1	0	1	1	EA = R + D Offset
1	x	x	1	1	1	0	0	EA = PC + 8 Bit Offset
1	x	x	1	1	1	0	1	EA = PC + 16 Bit Offset
1	R	R	1	1	1	1	1	EA = [Address]

Diagram labels:  
 Addressing Mode Field (bits 7-6)  
 Indirect Field (Sign bit when b7 = 0) (bits 5-4)  
 Register Field - RR (bits 3-2)

Legend:  
 x = Don't Care  
 d = Offset Bit  
 0 = Non Indirect  
 1 = Indirect  
 00 = X  
 01 = Y  
 10 = U  
 11 = S

300

HITACHI

Figure 16 Index Addressing Postbyte Register Bit Assignments



Table 2 Indexed Addressing Mode

Type	Forms	Non Indirect			Indirect		
		Assembler Form	Postbyte OP Code	+ ~ #	Assembler Form	Postbyte OP Code	+ ~ #
Constant Offset From R (2's Complement Offsets)	No Offset	,R	1RR00100	0 0	[,R]	1RR10100	3 0
	5 Bit Offset	n, R	0RRnnnnn	1 0	defaults to 8-bit		
	8 Bit Offset	n, R	1RR01000	1 1	[n, R]	1RR11000	4 1
	16 Bit Offset	n, R	1RR01001	4 2	[n, R]	1RR11001	7 2
Accumulator Offset From R (2's Complement Offsets)	A Register Offset	A, R	1RR00110	1 0	[A, R]	1RR10110	4 0
	B Register Offset	B, R	1RR00101	1 0	[B, R]	1RR10101	4 0
	D Register Offset	D, R	1RR01011	4 0	[D, R]	1RR11011	7 0
	Increment By 1	,R +	1RR00000	2 0	not allowed		
Auto Increment/Decrement R	Increment By 2	,R ++	1RR00001	3 0	[,R ++]	1RR10001	6 0
	Decrement By 1	,-R	1RR00010	2 0	not allowed		
	Decrement By 2	,--R	1RR00011	3 0	[,--R]	1RR10011	6 0
	8 Bit Offset	n, PCR	1xx01100	1 1	[n, PCR]	1xx11100	4 1
Constant Offset From PC (2's Complement Offsets)	16 Bit Offset	n, PCR	1xx01101	5 2	[n, PCR]	1xx11101	8 2
	Extended Indirect	16 Bit Address	—	—	[n]	10011111	5 2

R = X, Y, U or S      RR:  
 x = Don't Care      00 = X  
                           01 = Y  
                           10 = U  
                           11 = S

+ and # indicate the number of additional cycles and bytes for the particular variation.

**Zero-Offset Indexed**

In this mode, the selected pointer register contains the effective address of the data to be used by the instruction. This is the fastest indexing mode.

Examples are:  
 LDD 0, X  
 LDA S

**Constant Offset Indexed**

In this mode, a two's-complement offset and the contents of one of the pointer registers are added to form the effective address of the operand. The pointer register's initial content is unchanged by the addition.

Three sizes of offsets are available:  
 5-bit (-16 to +15)  
 8-bit (-128 to +127)  
 16-bit (-32768 to +32767)

The two's complement 5-bit offset is included in the postbyte and, therefore, is most efficient in use of bytes and cycles. The two's complement 8-bit offset is contained in a single byte following the postbyte. The two's complement 16-bit offset is in the two bytes following the postbyte. In most cases the programmer need not be concerned with the size of this offset since the assembler will select the optimal size automatically.

Examples of constant-offset indexing are:  
 LDA 23, X  
 LDX -2, S

LDY 300, X  
 LDU CAT, Y

**Accumulator-Offset Indexed**

This mode is similar to constant offset indexed except that the two's-complement value in one of the accumulators (A, B or D) and the contents of one of the pointer registers are added to form the effective address of the operand. The contents of both the accumulator and the pointer register are unchanged by the addition. The postbyte specifies which accumulator to use as an offset and no additional bytes are required. The advantage of an accumulator offset is that the value of the offset can be calculated by a program at run-time.

Some examples are:  
 LDA B, Y  
 LDX D, Y  
 LEAX B, X

**Auto Increment/Decrement Indexed**

In the auto increment addressing mode, the pointer register contains the address of the operand. Then, after the pointer register is used it is incremented by one or two. This addressing mode is useful in stepping through tables, moving data, or for the creation of software stacks. In auto decrement, the pointer register is decremented prior to use as the address of the data. The use of auto decrement is similar to that of auto increment; but the tables, etc., are scanned from the high to low addresses. The size of the increment/decrement can be either one or two to allow for tables of either 8- or 16-bit data to be accessed and is selectable by the programmer. The pre-

HD6809E, HD68A09E, HD68B09E

decrement, post-increment nature of these modes allow them to be used to create additional software stacks that behave identically to the U and S stacks.

Some examples of the auto increment/decrement addressing modes are:

```

LDA    ,X +
STD    ,Y ++
LDB    , - Y
LDX    , - - S
    
```

Care should be taken in performing operations on 16-bit pointer registers (X, Y, U, S) where the same register is used to calculate the effective address.

Consider the following instruction:

```
STX 0, X ++ (X initialized to 0)
```

The desired result is to store a 0 in locations \$0000 and \$0001 then increment X to point to \$0002. In reality, the following occurs:

```

0 → temp    calculate the EA; temp is a holding register
X + 2 → X   perform autoincrement
X → (temp)  do store operation
    
```

- Indexed Indirect**  
All of the indexing modes with the exception of auto increment/decrement by one, or a ±4-bit offset may have an additional level of indirection specified. In indirect addressing, the effective address is contained at the location specified by the contents of the Index Register plus any offset. In the example below, the A accumulator is loaded indirectly using an effective address calculated from the Index Register and an offset.

	Before Execution	
	A = XX (don't care)	
	X = \$F000	
\$0100	LDA [\$10, X]	EA is now \$F010
\$F010	\$F1	\$F150 is now the
\$F011	\$50	new EA
\$F150	\$AA	
	After Execution	
	A = \$AA (Actual Data Loaded)	
	X = \$F000	

All modes of indexed indirect are included except those which are meaningless (e.g., auto increment/decrement by 1 indirect). Some examples of indexed indirect are:

```

LDA    [, X]
LDD    [, 10, S]
LDA    [, Y]
LDD    [, X ++ ]
    
```

- Relative Addressing**  
The byte(s) following the branch opcode is (are) treated as a signed offset which may be added to the program counter. If the branch condition is true then the calculated address (PC + signed offset) is loaded into the program counter. Program execution continues at the new location as indicated by the PC; short (1 byte offset) and long (2 bytes offset) relative addressing modes are available. All of memory can be reached in long relative addressing as an effective address is interpreted modulo 2<sup>16</sup>. Some examples of relative addressing are:

BEQ	CAT	(short)
BGT	DOG	(short)

CAT	LBEQ	RAT	(long)
DOG	LBGT	RABBIT	(long)
	.	.	.
	.	.	.
RAT	NOP		
RABBIT	NOP		

- Program Counter Relative**  
The PC can be used as the pointer register with 8 or 16-bit signed offsets. As in relative addressing, the offset is added to the current PC to create the effective address. The effective address is then used as the address of the operand or data. Program Counter Relative Addressing is used for writing position independent programs. Tables related to a particular routine will maintain the same relationship after the routine is moved, if referenced relative to the Program Counter. Examples are:

```

LDA    CAT, PCR
LEAX   TABLE, PCR
    
```

Since program counter relative is a type of indexing, an additional level of indirection is available.

```

LDA    [CAT, PCR]
LDU    [DOG, PCR]
    
```

- HD6809E INSTRUCTION SET**  
The instruction set of the HD6809E is similar to that of the HD6800 and is upward compatible at the source code level. The number of opcodes has been reduced from 72 to 59, but because of the expanded architecture and additional addressing modes, the number of available opcodes (with different addressing modes) has risen from 197 to 1464. Some of the new instructions are described in detail below:

- PSHU/PSHS**  
The push instructions have the capability of pushing onto either the hardware stack (S) or user stack (U) any single register, or set of registers with a single instruction.

- PULU/PULS**  
The pull instructions have the same capability of the push instruction, in reverse order. The byte immediately following the push or pull opcode determines which register or registers are to be pushed or pulled. The actual PUSH/PULL sequence is fixed; each bit defines a unique register to push or pull, as shown in below.

PUSH/PULL POST BYTE

	← Pull Order		Push Order →				
PC	U	Y	X	DP	B	A	CC
FFFF .....	←	increasing memory address	.....	0000			
PC	S	Y	X	DP	B	A	CC

302

HITACHI

• **TFR/EXG**

Within the HD6809E, any register may be transferred to or exchanged with another of like-size; i.e., 8-bit to 8-bit or 16-bit to 16-bit. Bits 4~7 of postbyte define the source register, while bits 0~3 represent the destination register. These are denoted as follows:

0000 – D	0101 – PC
0001 – X	1000 – A
0010 – Y	1001 – B
0011 – U	1010 – CC
0100 – S	1011 – DP

(NOTE) All other combinations are undefined and INVALID.

**TRANSFER/EXCHANGE POST BYTE**



• **LEAX/LEAY/LEAU/LEAS**

The LEA (Load Effective Address) works by calculating the effective address used in an indexed instruction and stores that address value, rather than the data at that address, in a pointer register. This makes all the features of the internal addressing hardware available to the programmer. Some of the implications of this instruction are illustrated in Table 3.

The LEA instruction also allows the user to access data in a position independent manner. For example:

```
LEAX MSG1, PCR
LBSR PDATA (Print message routine)
*
*
MSG1 FCC 'MESSAGE'
```

This sample program prints: 'MESSAGE'. By writing MSG1, PCR, the assembler computes the distance between the present address and MSG1. This result is placed as a constant into the LEAX instruction which will be indexed from the PC value at the time of execution. No matter where the code is located, when it is executed, the computed offset from the PC will put the absolute address of MSG1 into the X pointer register. This code is totally position independent.

The LEA instructions are very powerful and use an internal holding register (temp). Care must be exercised when using the LEA instructions with the autoincrement and autodecrement addressing modes due to the sequence of internal operations. The LEA internal sequence is outlined as follows:

- LEAa, b+ (any of the 16-bit pointer registers X, Y, U or S may be substituted for a and b.)
1. b → temp (calculate the EA)
  2. b + 1 → b (modify b, postincrement)
  3. temp → a (load a)
- LEAa, – b
1. b – 1 → temp (calculate EA with predecrement)
  2. b – 1 → b (modify b, predecrement)
  3. temp → a (load a)

Autoincrement-by-two and autodecrement-by-two instructions work similarly. Note that LEAX, X+ does not change X, however LEAX, –X does decrement X. LEAX 1, X should be used to increment X by one.

Table 3 LEA Examples

Instruction	Operation	Comment
LEAX 10, X	X + 10 → X	Adds 5-bit constant 10 to X
LEAX 500, X	X + 500 → X	Adds 16-bit constant 500 to X
LEAY A, Y	Y + A → Y	Adds 8-bit A accumulator to Y
LEAY D, Y	Y + D → Y	Adds 16-bit D accumulator to Y
LEAU –10, U	U – 10 → U	Subtracts 10 from U
LEAS –10, S	S – 10 → S	Used to reserve area on stack
LEAS 10, S	S + 10 → S	Used to 'clean up' stack
LEAX 5, S	S + 5 → X	Transfers as well as adds

• **MUL**

Multiplies the unsigned binary numbers in the A and B accumulator and places the unsigned result into the 16-bit D accumulator. This unsigned multiply also allows multiple-precision multiplications.

**Long and Short Relative Branches**

The HD6809E has the capability of program counter relative branching throughout the entire memory map. In this mode, if the branch is to be taken, the 8 or 16-bit signed offset is added to the value of the program counter to be used as the effective address. This allows the program to branch anywhere in the 64k memory map. Position independent code can be easily generated through the use of relative branching. Both short (8-bit) and long (16-bit) branches are available.

• **SYNC**

After encountering a Sync instruction, the MPU enters a Sync state, stops processing instructions and waits for an interrupt. If the pending interrupt is non-maskable (NMI) or maskable (FIRQ, IRQ) with its mask bit (F or I) clear, the processor will clear the Sync state and perform the normal interrupt stacking and service routine. Since FIRQ and IRQ are not edge-triggered, a low level with a minimum duration of three bus cycles is required to assure that the interrupt will be taken. If the pending interrupt is maskable (FIRQ, IRQ) with its mask bit (F or I) set, the processor will clear the Sync state and continue processing by executing the next inline instruction. Figure 17 depicts Sync timing.

**Software Interrupts**

A Software Interrupt is an instruction which will cause an interrupt, and its associated vector fetch. These Software Interrupts are useful in operating system calls, software debugging, trace operations, memory mapping, and software development systems. Three levels of SWI are available on this HD6809E, and are prioritized in the following order: SWI, SWI2, SWI3.

**16-Bit Operation**

The HD6809E has the capability of processing 16-bit data. These instructions include loads, stores, compares, adds, subtracts, transfers, exchanges, pushes and pulls.

■ **CYCLE-BY-CYCLE OPERATION**

The address bus cycle-by-cycle performance chart illustrates the memory-access sequence corresponding to each possible instruction and addressing mode in the HD6809E. Each instruction begins with an opcode fetch. While that opcode is being internally decoded, the next program byte is always fetched. (Most instructions will use the next byte, so this

HD6809E, HD68A09E, HD68B09E

technique considerably speeds throughput.) Next, the operation of each opcode will follow the flow chart.  $\overline{VMA}$  is an indication of  $FFFF_{16}$  on the address bus,  $R/\overline{W}$  = "High" and  $BS$  = "Low". The following examples illustrate the use of the chart; see Figure 18.

Example 1: LBSR (Branch Taken)  
Before Execution  $SP = F000$

	.		
	.		
	.		
\$8000	LBSR	CAT	
	.		
	.		
	.		
SA000	CAT		
	.		

CYCLE-BY-CYCLE FLOW

Cycle #	Address	Data	R/ $\overline{W}$	Description
1	8000	17	1	Opcode Fetch
2	8001	1F	1	Offset High Byte
3	8002	FD	1	Offset Low Byte
4	FFFF	*	1	VMA Cycle
5	FFFF	*	1	VMA Cycle
6	FFFF	*	1	VMA Cycle
7	FFFF	*	1	VMA Cycle
8	FFFF	03	0	Stack Low Order Byte of Return Address
9	FFFE	80	0	Stack High Order Byte of Return Address

Example 2: DEC (Extended)

\$8000	DEC	\$A000
SA000	FCB	\$80

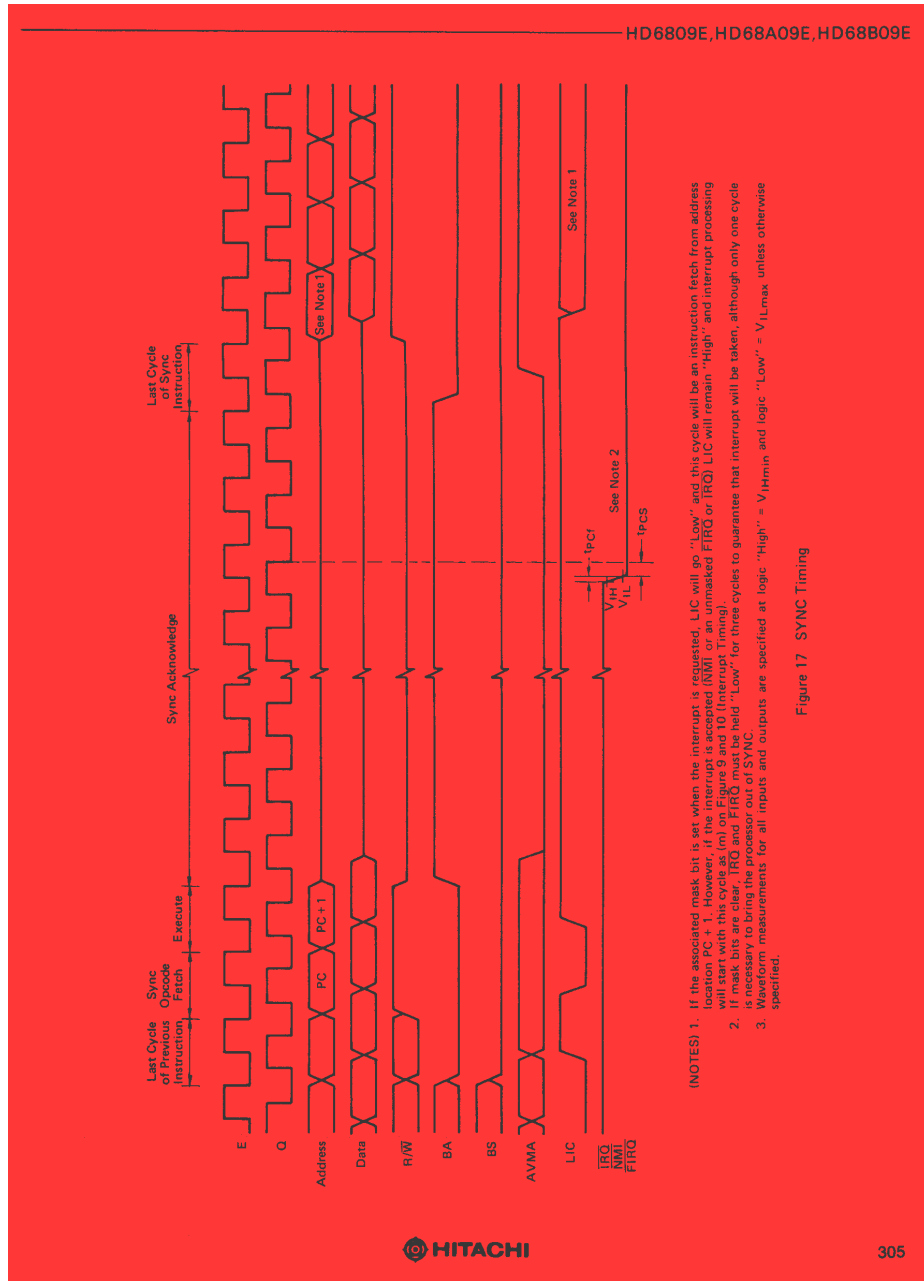
Cycle #	Address	Data	R/ $\overline{W}$	Description
1	8000	7A	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	VMA Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	VMA Cycle
7	A000	7F	0	Store the Decre- mented Data

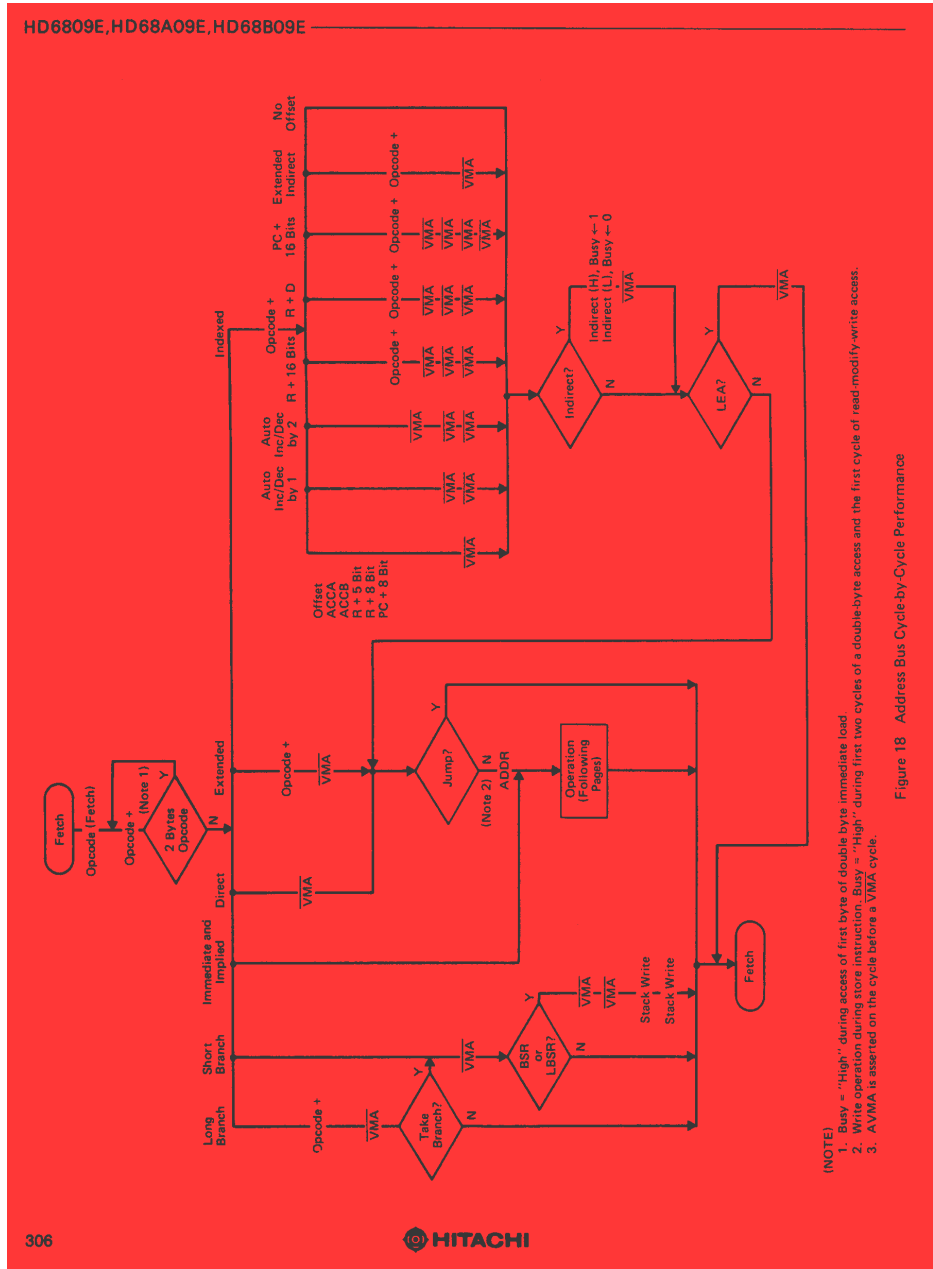
\* The data bus has the data at that particular address.

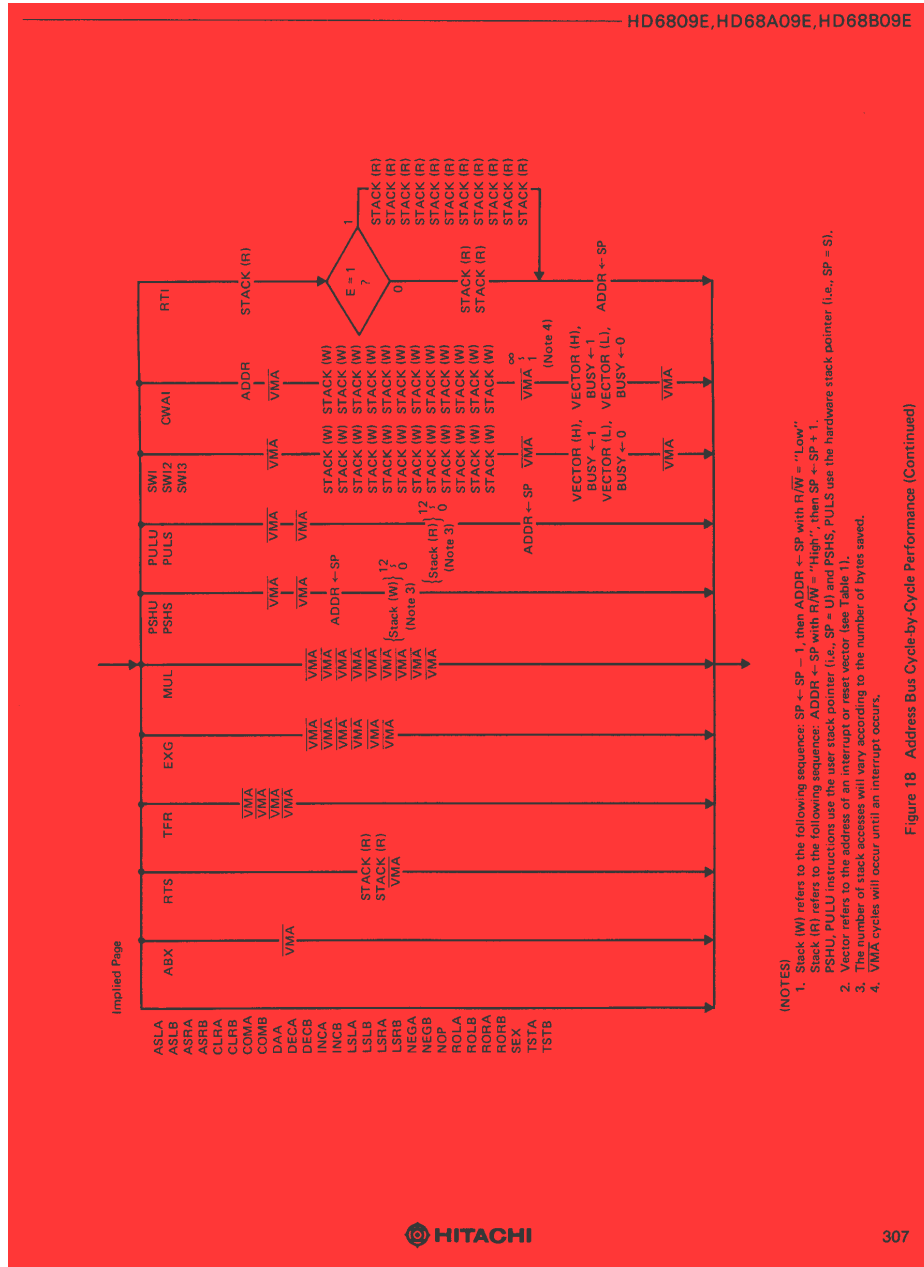
■ HD6809E INSTRUCTION SET TABLES

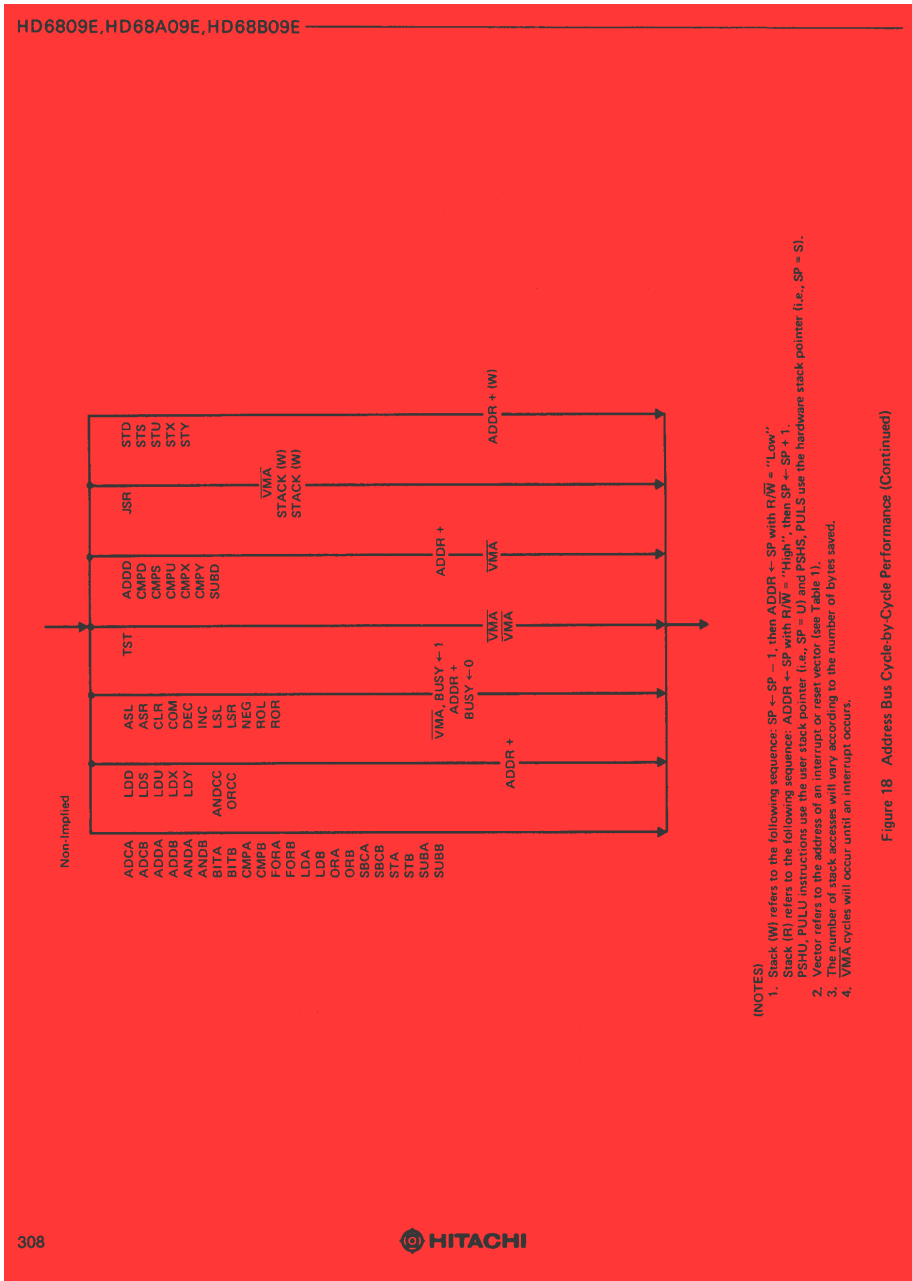
The instructions of the HD6809E have been broken down into five different categories. They are as follows:  
8-Bit operation (Table 4)  
16-Bit operation (Table 5)  
Index register/stack pointer instructions (Table 6)  
Relative branches (long or short) (Table 7)  
Miscellaneous instructions (Table 8)

HD6809E instruction set tables and Hexadecimal Values of instructions are shown in Table 9 and Table 10.











HD6809E, HD68A09E, HD68B09E

Table 4 8-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADCA, ADCB	Add memory to accumulator with carry
ADDA, ADDB	Add memory to accumulator
ANDA, ANDB	And memory with accumulator
ASL, ASLA, ASLB	Arithmetic shift of accumulator or memory left
ASR, ASRA, ASRB	Arithmetic shift of accumulator or memory right
BITA, BITB	Bit test memory with accumulator
CLR, CLRA, CLRB	Clear accumulator or memory location
CMPA, CMPB	Compare memory from accumulator
COM, COMA, COMB	Complement accumulator or memory location
DAA	Decimal adjust A accumulator
DEC, DECA, DECB	Decrement accumulator or memory location
EORA, EORB	Exclusive or memory with accumulator
EXG R1, R2	Exchange R1 with R2 (R1, R2 = A, B, CC, DP)
INC, INCA, INCB	Increment accumulator or memory location
LDA, LDB	Load accumulator from memory
LSL, LSLA, LSLB	Logical shift left accumulator or memory location
LSR, LSRA, LSRB	Logical shift right accumulator or memory location
MUL	Unsigned multiply (A × B → D)
NEG, NEGA, NEGB	Negate accumulator or memory
ORA, ORB	Or memory with accumulator
ROL, ROLA, ROLB	Rotate accumulator or memory left
ROR, RORA, RORB	Rotate accumulator or memory right
SBCA, SBCB	Subtract memory from accumulator with borrow
STA, STB	Store accumulator to memory
SUBA, SUBB	Subtract memory from accumulator
TST, TSTA, TSTB	Test accumulator or memory location
TFR R1, R2	Transfer R1 to R2 (R1, R2 = A, B, CC, DP)

(NOTE) A, B, CC or DP may be pushed to (pulled from) either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 5 16-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADDD	Add memory to D accumulator
CMPD	Compare memory from D accumulator
EXG D, R	Exchange D with X, Y, S, U or PC
LDD	Load D accumulator from memory
SEX	Sign Extend B accumulator into A accumulator
STD	Store D accumulator to memory
SUBD	Subtract memory from D accumulator
TFR D, R	Transfer D to X, Y, S, U or PC
TFR R, D	Transfer X, Y, S, U or PC to D

(NOTE) D may be pushed (pulled) to either stack with PSHS, PSHU (PULS, PULU) instructions.

HD6809E, HD68A09E, HD68B09E

Table 6 Index Register Stack Pointer Instructions

Mnemonic(s)	Operation
CMPS, CMPU	Compare memory from stack pointer
CMPX, CMPY	Compare memory from index register
EXG R1, R2	Exchange D, X, Y, S, U or PC with D, X, Y, S, U or PC
LEAS, LEAU	Load effective address into stack pointer
LEAX, LEAY	Load effective address into index register
LDS, LDU	Load stack pointer from memory
LDX, LDY	Load index register from memory
PSHS	Push A, B, CC, DP, D, X, Y, U, or PC onto hardware stack
PSHU	Push A, B, CC, DP, D, X, Y, S, or PC onto user stack
PULS	Pull A, B, CC, DP, D, X, Y, U or PC from hardware stack
PULU	Pull A, B, CC, DP, D, X, Y, S or PC from user stack
STS, STU	Store stack pointer to memory
STX, STY	Store index register to memory
TFR R1, R2	Transfer D, X, Y, S, U or PC to D, X, Y, S, U or PC
ABX	Add B accumulator to X (unsigned)

Table 7 Branch Instructions

Mnemonic(s)	Operation
<b>SIMPLE BRANCHES</b>	
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BMI, LBMI	Branch if minus
BPL, LBPL	Branch if plus
BCS, LBCS	Branch if carry set
BCC, LBCC	Branch if carry clear
BVS, LBVS	Branch if overflow set
BVC, LBVC	Branch if overflow clear
<b>SIGNED BRANCHES</b>	
BGT, LBGT	Branch if greater (signed)
BGE, LBGE	Branch if greater than or equal (signed)
BEQ, LBEQ	Branch if equal
BLE, LBLE	Branch if less than or equal (signed)
BLT, LBLT	Branch if less than (signed)
<b>UNSIGNED BRANCHES</b>	
BHI, LBHI	Branch if higher (unsigned)
BHS, LBHS	Branch if higher or same (unsigned)
BEQ, LBEQ	Branch if equal
BLS, LBLS	Branch if lower or same (unsigned)
BLO, LBLO	Branch if lower (unsigned)
<b>OTHER BRANCHES</b>	
BSR, LBSR	Branch to subroutine
BRA, LBRA	Branch always
BRN, LBRN	Branch never

HD6809E,HD68A09E,HD68B09E

Table 8 Miscellaneous Instructions

Mnemonic(s)	Operation
ANDCC	AND condition code register
CWAI	AND condition code register, then wait for interrupt
NOP	No operation
ORCC	OR condition code register
JMP	Jump
JSR	Jump to subroutine
RTI	Return from interrupt
RTS	Return from subroutine
SWI, SWI2, SWI3	Software interrupt (absolute indirect)
SYNC	Synchronize with interrupt line

HD6809E, HD68A09E, HD68B09E

Table 9 HD6809E Instruction Set Table

INSTRUCTION/ FORMS	HD6809E ADDRESSING MODES												DESCRIPTION	H	N	Z	V	C							
	IMPLIED			DIRECT			EXTENDED			IMMEDIATE									INDEXED <sup>Q</sup>			RELATIVE			
	OP	~	#	OP	~	#	OP	~	#	OP	~	#							OP	~	#	OP	~	#	
ABX	3A	3	1																$R(X) \leftarrow X$ (UNSIGNED)	•	•	•	•	•	
ADC	ADCA ADCB			99 D9	4 4	2 2	B9 F9	5 5	3 3	89 C9	2 2	2 2	A9 E9	4+ 4+	2+ 2+				$A \leftarrow A + M$ $B \leftarrow B + C$	•	•	•	•	•	
ADD	ADDA ADDB ADDD			9B DB D3	4 4 6	2 2 2	BB FB F3	5 5 7	3 3 3	8B CB C3	2 2 4	2 2 3	AB EB E3	4+ 4+ 6+	2+ 2+ 2+				$A \leftarrow A + M$ $B \leftarrow B + M$ $D \leftarrow D + M + 1$	•	•	•	•	•	
AND	ANDA ANDB ANDCC			94 D4	4 4	2 2	B4 F4	5 5	3 3	84 C4 1C	2 2 3	2 2 3	A4 E4	4+ 4+	2+ 2+				$A \leftarrow A \wedge M$ $B \leftarrow B \wedge M$ $CC \leftarrow CC \wedge IMM$	•	•	•	•	•	
ASL	ASLA ASLB ASL	48 58	2 2	1 1														$A \leftarrow A \ll 1$ $B \leftarrow B \ll 1$ $M \leftarrow M \ll 1$	•	•	•	•	•		
ASR	ASRA ASRB ASR	47 57	2 2	1 1	08	6	2	78	7	3			68	6+	2+				$A \leftarrow A \gg 1$ $B \leftarrow B \gg 1$ $M \leftarrow M \gg 1$	•	•	•	•	•	
BCC	BCC LBCC															24 10 24	3 5(6) 4	2 4	Branch C = 0 Long Branch C = 0	•	•	•	•	•	
BCS	BCS LBCS															25 10 25	3 5(6) 4	2 4	Branch C = 1 Long Branch C = 1	•	•	•	•	•	
BEQ	BEQ LBEQ															27 10 27	3 5(6) 4	2 4	Branch Z = 1 Long Branch Z = 1	•	•	•	•	•	
BGE	BGE LBGE															2C 10 2C	3 5(6) 4	2 4	Branch N @ V = 0 Long Branch N @ V = 0	•	•	•	•	•	
BGT	BGT LBGT															2E 10 2E	3 5(6) 4	2 4	Branch Z @ (N @ V) = 0 Long Branch Z @ (N @ V) = 0	•	•	•	•	•	
BHI	BHI LBHI															22 10 22	3 5(6) 4	2 4	Branch CV Z = 0 Long Branch CV Z = 0	•	•	•	•	•	
BHS	BHS LBHS															24 10 24	3 5(6) 4	2 4	Branch C = 0 Long Branch C = 0	•	•	•	•	•	
BIT	BITA BITB				95 D5	4 4	2 2	B5 F5	5 5	3 3	85 C5	2 2	2 2	A5 E5	4+ 4+	2+ 2+				Bit Test A (M ^ A) Bit Test B (M ^ B)	•	•	•	•	•
BLE	BLE LBLE															2F 10 2F	3 5(6) 4	2 4	Branch Z @ (N @ V) = 1 Long Branch Z @ (N @ V) = 1	•	•	•	•	•	
BLO	BLO LBLO															25 10 25	3 5(6) 4	2 4	Branch C = 1 Long Branch C = 1	•	•	•	•	•	
BLS	BLS LBLS															23 10 23	3 5(6) 4	2 4	Branch CV Z = 1 Long Branch CV Z = 1	•	•	•	•	•	
BLT	BLT LBLT															2D 10 2D	3 5(6) 4	2 4	Branch N @ V = 1 Long Branch N @ V = 1	•	•	•	•	•	
BMI	BMI LBMI															28 10 28	3 5(6) 4	2 4	Branch N = 1 Long Branch N = 1	•	•	•	•	•	
BNE	BNE LBNE															26 10 26	3 5(6) 4	2 4	Branch Z = 0 Long Branch Z = 0	•	•	•	•	•	
BPL	BPL LBPL															2A 10 2A	3 5(6) 4	2 4	Branch N = 0 Long Branch N = 0	•	•	•	•	•	
BRA	BRA LBRA															20 16 5	3 5 3	2 3	Branch Always Long Branch/ Always	•	•	•	•	•	
BRN	BRN LBRN															21 10 21	3 5 4	2 4	Branch Never Long Branch Never	•	•	•	•	•	

(to be continued)

		HD6809E ADDRESSING MODES													DESCRIPTION	S	3	2	1	0					
INSTRUCTION/ FORMS		IMPLIED			DIRECT			EXTENDED			IMMEDIATE			INDEXED <sup>1</sup>							RELATIVE				
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	5	H	N	Z	V	C
BSR	BSR																		8D	Branch to Subroutine					
	LBSR																		17	Long Branch to Subroutine					
BVC	BVC																		28	Branch V = 0					
	LBVC																		10	Long Branch V = 0					
BVS	BVS																		28	Branch V = 1					
	LBVS																		10	Long Branch V = 1					
CLR	CLRA	4F	2	1															0 → A						
	CLRB	5F	2	1															0 → B						
CMP	CMPA				0F	6	2	7F	7	3						6F	6+	2+		Compare M from A					
	CMPB				D1	4	2	F1	5	3	C1	2	2			A1	4+	2+		Compare M from B					
CMPD	CMPD				10	7	3	10	8	4	10	5	4			10	7+	3+		Compare M: M + 1 from D					
	CMPS				11	7	3	11	8	4	11	5	4			A3	7+	3+		Compare M: M + 1 from S					
CMPI	CMPI				9C			BC	8	4	11	5	4			AC	7+	3+		Compare M: M + 1 from U					
	CMPI				11	7	3	11	8	4	11	5	4			A3	7+	3+		Compare M: M + 1 from X					
CMPX	CMPX				93	6	2	BC	7	3	8C	4	3			AC	6+	2+		Compare M: M + 1 from X					
	CMPY				9C	7	3	10	8C	8	4	10	8C			10	7+	3+		Compare M: M + 1 from Y					
COM	COMA	43	2	1															$\bar{A} \rightarrow A$						
	COMB	53	2	1															$\bar{B} \rightarrow B$						
CWA1	CWA1	3C	20	2															$\bar{M} \rightarrow M$						
	CWA1				03	6	2	73	7	3						63	6+	2+		CC A IMM - CC (except 1-E)					
DAA	DAA	19	2	1															Wait for Interrupt						
DEC	DECA	4A	2	1															Decimal Adjust A						
	DECB	5A	2	1															Decimal Adjust B						
EOR	EORA				0A	6	2	7A	7	3						6A	6+	2+		A ⊕ M → A					
	EORB				98	4	2	B8	5	3	88	2	2			A8	4+	2+		B ⊕ M → B					
EXG	EXG	1E	7	2															R1 → R2, R2 → R1						
	EXG				D8	4	2	F8	5	3	C8	2	2			E8	4+	2+		R1 → R2, R2 → R1					
INC	INCA	4C	2	1															A + 1 → A						
	INCB	5C	2	1															B + 1 → B						
JMP	JMP				0C	6	2	7C	7	3						6C	6+	2+		M + 1 → M					
	JMP				0E	3	2	7E	4	3						6E	3+	2+		EA <sup>1</sup> → PC					
JSR	JSR				9D	7	2	BD	8	3						AD	7+	2+		Jump to Subroutine					
	LD	LDA				96	4	2	B6	5	3	86	2	2		A6	4+	2+		M → A					
LD	LDB				D6	4	2	F6	5	3	C6	2	2		E6	4+	2+		M → B						
	LDD				DC	5	2	FC	6	3	CC	3	3		EC	5+	2+		M: M + 1 → D						
LD	LDS				10	6	3	10	7	4	10	6+	3+		10	6+	3+		M: M + 1 → S						
	LDU				DE	5	2	FE	6	3	CE	3	3		EE	5+	2+		M: M + 1 → U						
LD	LDX				9E	5	2	BE	6	3	8E	3	3		AE	5+	2+		M: M + 1 → X						
	LDY				10	6	3	10	7	4	10	6+	3+		10	6+	3+		M: M + 1 → Y						
LEA	LEAS				9E			BE	7	4	8E					AE				EA <sup>3</sup> → S					
	LEAU																			EA <sup>3</sup> → U					
	LEAX																			EA <sup>3</sup> → X					
	LEAY																			EA <sup>3</sup> → Y					
LSL	LSLA	48	2	1															A1						
	LSLB	58	2	1															B1						
LSR	LSRA	44	2	1	08	6	2	78	7	3					68	6+	2+		M1						
	LSRB	54	2	1															A1						
MUL	MUL	3D	11	1															B1						
	MUL				04	6	2	74	7	3					64	6+	2+		M1						
NEG	NEGA	40	2	1															$\bar{A} + 1 \rightarrow A$						
	NEGB	50	2	1															$\bar{B} + 1 \rightarrow B$						
NOP	NEGA				00	6	2	70	7	3					60	6+	2+		M + 1 → M						
	NEGB																		No Operation						

(to be continued)

INSTRUCTION/ FORMS		HD6809E ADDRESSING MODES												DESCRIPTION	5 3 2 1 0																	
		IMPLIED			DIRECT			EXTENDED			IMMEDIATE				INDEXED <sup>①</sup>		RELATIVE		H	N	Z	V	C									
		OP	~	#	OP	~	#	OP	~	#	OP	~	#		OP	~	OP	~														
OR	ORA ORB ORCC				9A DA	4 4	2 2				BA FA	5 5	3 3				8A CA 1A	2 2 3	2 2 2	AA EA	4+ 4+	2+ 2+						A ← V ← M → A B ← V ← M → B CC ← V ← IMM → CC	• 1 1 0 • • 1 1 0 • (← 7)			
PSH	PSHS PSHU	34	5+ <sup>⑤</sup>	2																									Push Registers on S Stack Push Registers on U Stack	• • • • • • • • • •		
		35	5+ <sup>⑤</sup>	2																									Pull Registers from S Stack Pull Registers from U Stack	(← 18) (← 10)		
ROL	ROLA ROLB ROL	49	2	1																										A) • 1 : : : : : B) • 1 : : : : : M) • 1 : : : : :	Return From Interrupt	(← 7)
		59	2	1	09	6	2	79	7	3				69	6+	2+																
ROR	RORA RORB ROR	46	2	1																										A) • 1 : : : : : B) • 1 : : : : : M) • 1 : : : : :	Return From Subroutine	(← 7)
		56	2	1	06	6	2	76	7	3				66	6+	2+																
RTI		38	6/15	1																									Return From Interrupt	(← 7)		
RTS		39	5	1																									Return From Subroutine	• • • • •		
SBC	SBCA SBCB				92 D2	4 4	2 2	B2 F2	5 5	3 3				82 C2	2 2	2 2	A2 E2	4+ 4+	2+ 2+											A ← M ← C → A B ← M ← C → B	8 : 1 1 : : 1 8 : 1 1 : : 1	
		SEX	10	2	1																								Sign Extend B into A	• 1 1 • •		
ST	STA STB STD STS  STU STX STY				97 D7 DD	4 4 5	2 2 2	87 F7 FD	5 5 6	3 3 3							A7 E7 ED	4+ 4+ 5+	2+ 2+ 2+												A ← M B ← M D ← M: M + 1 S ← M: M + 1	• 1 1 0 • • 1 1 0 • • 1 1 0 • • 1 1 0 •
					10	6	3	10	7	4							10	6+	3+											U ← M: M + 1 X ← M: M + 1 Y ← M: M + 1	• 1 : : 0 • • 1 : : 0 • • 1 : : 0 •	
					9F	5	2	BF	6	3							AF	5+	2+													
					10	6	3	10	7	4							10	6+	3+													
					9F	5	2	BF	6	3							AF	5+	2+													
					9F	5	2	BF	6	3							AF	5+	2+													
					9F	5	2	BF	6	3							AF	5+	2+													
SUB	SUBA SUBB SUBD				90 D0	4 4	2 2	80 F0	5 5	3 3	80 C0	2 2	2 2	A0 E0	4+ 4+	2+ 2+													A ← M → A B ← M → B D ← M: M + 1 → D	8 : 1 1 : : 1 8 : 1 1 : : 1 • : : : 1 1 :		
		SWI	3F	19	1																								Software Interrupt1	• • • • •		
			10	20	2																								Software Interrupt2	• • • • •		
						3F	11	20	2																			Software Interrupt3	• • • • •			
SYNC		13	>2	1																								Synchronize to Interrupt	• • • • •			
TFR	R1, R2	1F	6	2																								R1 → R2 <sup>②</sup>	(← 10)			
TST	TSTA TSTB TST	4D	2	1																								Test A	• 1 : : 0 •			
		5D	2	1																								Test B	• : : : 0 •			
					0D	6	2	7D	7	3				6D	6+	2+												Test M	• 1 : : 0 •			

(NOTES)

- ① This column gives a base cycle and byte count. To obtain total count, and the values obtained from the INDEXED ADDRESSING MODES table.
- ② R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.
- ③ The 8 bit registers are: A, B, CC, DP.
- ④ The 16 bit registers are: X, Y, U, S, D, PC.
- ⑤ EA is the effective address.
- ⑥ The PSH and PUL instructions require 5 cycle plus 1 cycle for each byte pushed or pulled.
- ⑦ S(6) means: 5 cycles if branch not taken, 6 cycles if taken.
- ⑧ SWI sets 1 and F bits. SWI2 and SWI3 do not affect 1 and F.
- ⑨ Conditions Codes set as a direct result of the instruction.
- ⑩ Value of half-carry flag is undefined.
- ⑪ Special Case — Carry set if b7 is SET.
- ⑫ Condition Codes set as a direct result of the instruction if CC is specified, and not affected otherwise.

LEGEND:

- OP Operation Code (Hexadecimal)
- ~ Number of MPU Cycles
- # Number of Program Bytes
- + Arithmetic Plus
- Arithmetic Minus
- x Multiply
- M Complement of M
- Transfer Into
- H Half-carry (from bit 3)
- N Negative (sign bit)
- Z Zero (byte)
- V Overflow, 2's complement
- C Carry from bit 7
- ↑ Test and set if true, cleared otherwise
- Not Affected
- CC Condition Code Register
- ⋮ Concatenation
- ∨ Logical or
- ∧ Logical and
- ⊕ Logical Exclusive or




HD6809E, HD68A09E, HD68B09E

Table 10 Hexadecimal Values of Machine Codes

OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
00	NEG	Direct	6	2	30	LEAX	Indexed	4+	2+	60	NEG	Indexed	6+	2+
01	*	↑	6	2	31	LEAY	Indexed	4+	2+	61	*	↑	6+	2+
02	*				32	LEAS				62	*			
03	COM	6	2	2	33	LEAU	Indexed	4+	2+	63	COM	↑	6+	2+
04	LSR				34	PSHS				64	LSR			
05	*	6	2	2	35	PULS	↑	5+	2	65	*	↑	6+	2+
06	ROR				36	PSHU				66	ROR			
07	ASR	6	2	2	37	PULU	↑	5+	2	67	ASR	↑	6+	2+
08	ASL, LSL				38	*				68	ASL, LSL			
09	ROL	6	2	2	39	RTS	5	1	1	69	ROL	↑	6+	2+
0A	DEC				3A	ABX				6A	DEC			
0B	*	6	2	2	3B	RTI	6, 15	1	1	6B	*	↑	6+	2+
0C	INC				3C	QWAI				6C	INC			
0D	TST	6	2	2	3D	MUL	20	2	2	6D	TST	↑	6+	2+
0E	JMP				3E	*				6E	JMP			
0F	CLR	Direct	6	2	3F	SWI	Implied	19	1	6F	CLR	Indexed	6+	2+
10	} See Next Page	—	—	—	40	NEGA	Implied	2	1	70	NEG	Extended	7	3
11		—	—	—	41	*	↑	2	1	71	*	↑	7	3
12	NOP	Implied	2	1	42	*				2	1			
13	SYNC	Implied	2	1	43	COMA	↑	2	1			73	COM	↑
14	*				44	LSRA				74	LSR			
15	*	Relative	5	3	45	*	↑	2	1	75	*	↑	7	3
16	LBRA				46	RORA				76	ROR			
17	LBSR	Relative	9	3	47	ASRA	↑	2	1	77	ASR	↑	7	3
18	*				48	ASLA, LSLA				78	ASL, LSL			
19	DAA	Implied	2	1	49	ROLA	↑	2	1	79	ROL	↑	7	3
1A	ORCC	Immed	3	2	4A	DECA				7A	DEC			
1B	*	—	—	—	4B	*	↑	2	1	7B	*	↑	7	3
1C	ANDCC	Immed	3	2	4C	INCA				7C	INC			
1D	SEX	Implied	2	1	4D	TSTA	↑	2	1	7D	TST	↑	7	3
1E	EXG	8	2	4E	*	7E				JMP				
1F	TFR	Implied	6	2	4F	CLRA	Implied	2	1	7F	CLR	Extended	7	3
20	BRA	Relative	3	2	50	NEGB	↑	2	1	80	SUBA	↑	2	2
21	BRN	3	2	51	*	81				CMPA				
22	BHI	3	2	2	52	*	↑	2	1	82	SBCA	↑	2	2
23	BLS				53	COMB				83	SUBD			
24	BHS, BCC	3	2	2	54	LSRB	↑	2	1	84	ANDA	↑	2	2
25	BLO, BCS				55	*				85	BITA			
26	BNE	3	2	2	56	RORB	↑	2	1	86	LDA	↑	2	2
27	BEQ				57	ASRB				87	*			
28	BVC	3	2	2	58	ASLB, LSLB	↑	2	1	88	EORA	↑	2	2
29	BVS				59	ROLB				89	ADCA			
2A	BPL	3	2	2	5A	DECB	↑	2	1	8A	ORA	↑	2	2
2B	BMI				5B	*				8B	ADDA			
2C	BGE	3	2	2	5C	INCB	↑	2	1	8C	CMPX	↑	4	3
2D	BLT				5D	TSTB				8D	BSR			
2E	BGT	3	2	2	5E	*	↑	2	1	8E	LDX	↑	3	3
2F	BLE				Relative	3				2	5F			

LEGEND: ~ Number of MPU cycles (less possible push pull or indexed-mode cycles)  
 # Number of program bytes  
 \* Denotes unused opcode

(to be continued)

 HITACHI

315

HD6809E, HD68A09E, HD68B09E														
OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
90	SUBA	Direct	4	2	C6	LDB	Immed	2	2	FC	LDD	Extended	6	3
91	CMPA	↑	4	2	C7	*				FD	STD	↑	6	3
92	SBCA	↑	4	2	C8	EORB	↑	2	2	FE	LDU	↓	6	3
93	SUBD	↑	6	2	C9	ADCB	↑	2	2	FF	STU	↓	6	3
94	ANDA	↑	4	2	CA	ORB	↑	2	2	2 Bytes Opcode				
95	BITA	↑	4	2	C8	ADDB	↑	2	2					
96	LDA	↑	4	2	CC	LDD	↓	3	3	1021	LBRN	Relative	5	4
97	STA	↑	4	2	CD	*				1022	LBHI	↑	5(6)	4
98	EORA	↑	4	2	CE	LDU	Immed	3	3	1023	LBLS	↑	5(6)	4
99	ADCA	↑	4	2	CF	*				1024	LBHS, LBCC	↑	5(6)	4
9A	ORA	↑	4	2						1025	LBGS, LBLO	↑	5(6)	4
9B	ADDA	↑	4	2	D0	SUBB	Direct	4	2	1026	LBNE	↑	5(6)	4
9C	CMPX	↑	6	2	D1	CMPB	↑	4	2	1027	LBEG	↑	5(6)	4
9D	JSR	↑	7	2	D2	SBCB	↑	4	2	1028	LBVC	↑	5(6)	4
9E	LDX	↑	5	2	D3	ADDD	↑	6	2	1029	LBVS	↑	5(6)	4
9F	STX	Direct	5	2	D4	ANDB	↑	4	2	102A	LBPL	↑	5(6)	4
					D5	BITB	↑	4	2	102B	LBMI	↑	5(6)	4
A0	SUBA	Indexed	4+	2+	D6	LDB	↑	4	2	102C	LBGE	↑	5(6)	4
A1	CMPA	↑	4+	2+	D7	STB	↑	4	2	102D	LBLT	↑	5(6)	4
A2	SBCA	↑	4+	2+	D8	EORB	↑	4	2	102E	LBGT	↑	5(6)	4
A3	SUBD	↑	6+	2+	D9	ADCB	↑	4	2	102F	LBLE	Relative	5(6)	4
A4	ANDA	↑	4+	2+	DA	ORB	↑	4	2	103F	SWI2	Implied	20	2
A5	BITA	↑	4+	2+	DB	ADDB	↑	4	2	1083	CMPD	Immed	5	4
A6	LDA	↑	4+	2+	DC	LDD	↓	5	2	108C	CMPY	↑	5	4
A7	STA	↑	4+	2+	DD	STD	↑	5	2	108E	LDY	Immed	4	4
A8	EORA	↑	4+	2+	DE	LDU	Direct	5	2	1093	CMPD	Direct	7	3
A9	ADCA	↑	4+	2+	DF	STU	↑	5	2	109C	CMPY	↑	7	3
AA	ORA	↑	4+	2+						109E	LDY	↑	6	3
AB	ADDA	↑	4+	2+	E0	SUBB	Indexed	4+	2+	109F	STY	Direct	6	3
AC	CMPX	↑	6+	2+	E1	CMPB	↑	4+	2+	10A3	CMPD	Indexed	7+	3+
AD	JSR	↑	7+	2+	E2	SBCB	↑	4+	2+	10AC	CMPY	↑	7+	3+
AE	LDX	↑	5+	2+	E3	ADDD	↑	6+	2+	10AE	LDY	↑	6+	3+
AF	STX	Indexed	5+	2+	E4	ANDB	↑	4+	2+	10AF	STY	Indexed	6+	3+
					E5	BITB	↑	4+	2+	10B3	CMPD	↑	8	4
B0	SUBA	Extended	5	3	E6	LDB	↑	4+	2+	10BC	CMPY	↑	8	4
B1	CMPA	↑	5	3	E7	STB	↑	4+	2+	10BE	LDY	↑	7	4
B2	SBCA	↑	5	3	E8	EORB	↑	4+	2+	10BF	STY	Extended	7	4
B3	SUBD	↑	7	3	E9	ADCB	↑	4+	2+	10CE	LDS	Immed	4	4
B4	ANDA	↑	5	3	EA	ORB	↑	4+	2+	10DE	LDS	Direct	6	3
B5	BITA	↑	5	3	EB	ADDB	↑	4+	2+	10DF	STS	Direct	6	3
B6	LDA	↑	5	3	EC	LDD	↓	5+	2+	10EE	LDS	Indexed	6+	3+
B7	STA	↑	5	3	ED	STD	↑	5+	2+	10EF	STS	Indexed	6+	3+
B8	EORA	↑	5	3	EE	LDU	Indexed	5+	2+	10FE	LDS	Extended	7	4
B9	ADCA	↑	5	3	EF	STU	↑	5+	2+	10FF	STS	Extended	7	4
BA	ORA	↑	5	3						113F	SWI3	Implied	20	2
BB	ADDA	↑	5	3	F0	SUBB	Extended	5	3	1183	CMPU	Immed	5	4
BC	CMPX	↑	7	3	F1	CMPB	↑	5	3	118C	CMPS	Immed	5	4
BD	JSR	↑	8	3	F2	SBCB	↑	5	3	1193	CMPU	Direct	7	3
BE	LDX	↑	6	3	F3	ADDD	↑	7	3	119C	CMPS	Direct	7	3
BF	STX	Extended	6	3	F4	ANDB	↑	5	3	11A3	CMPU	Indexed	7+	3+
					F5	BITB	↑	5	3	11AC	CMPS	Indexed	7+	3+
C0	SUBB	Immed	2	2	F6	LDB	↑	5	3	11B3	CMPU	Extended	8	4
C1	CMPB	↑	2	2	F7	STB	↑	5	3	11BC	CMPS	Extended	8	4
C2	SBCB	↑	2	2	F8	EORB	↑	5	3					
C3	ADDD	↑	4	3	F9	ADCB	↑	5	3					
C4	ANDB	↑	2	2	FA	ORB	↑	5	3					
C5	BITB	Immed	2	2	FB	ADDB	Extended	5	3					

(NOTE): All unused opcodes are both undefined and illegal



■ **NOTE FOR USE**

**Execution Sequence of CLR Instruction**

Cycle-by-cycle flow of CLR instruction (Direct, Extended, Indexed Addressing Mode) is shown below. In this sequence the content of the memory location specified by the operand is read before writing "00" into it. Note that status Flags, such as IRQ Flag, will be cleared by this extra data read operation when accessing the control/status register (sharing the same address between read and write) of peripheral devices.

**Example: CLR (Extended)**

\$8000 CLR \$A000  
\$A000 FCB \$80

Cycle #	Address	Data	R/W	Description
1	8000	7F	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	VMA Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	VMA Cycle
7	A000	00	0	Store Fixed "00" into Specified Location

\* The data bus has the data at that particular address.



## **C.8 AD7524 Analog/Digital Converter**





**CMOS**  
**8-Bit Buffered Multiplying DAC**

**AD7524**

**FEATURES**

- Microprocessor Compatible (6800, 8085, Z80, Etc.)
- TTL/CMOS Compatible Inputs
- On-Chip Data Latches
- Endpoint Linearity
- Low Power Consumption
- Monotonicity Guaranteed (Full Temperature Range)
- Latch Free (No Protection Schottky Required)

**APPLICATIONS**

- Microprocessor Controlled Gain Circuits
- Microprocessor Controlled Attenuator Circuits
- Microprocessor Controlled Function Generation
- Precision AGC Circuits
- Bus Structured Instruments

**GENERAL DESCRIPTION**

The AD7524 is a low cost, 8-bit monolithic CMOS DAC designed for direct interface to most microprocessors.

Basically an 8-bit DAC with input latches, the AD7524's load cycle is similar to the "write" cycle of a random access memory. Using an advanced thin-film on CMOS fabrication process, the AD7524 provides accuracy to 1/8 LSB with a typical power dissipation of less than 10 milliwatts.

A newly improved design eliminates the protection Schottky previously required and guarantees TTL compatibility when using a +5 V supply. Loading speed has been increased for compatibility with most microprocessors.

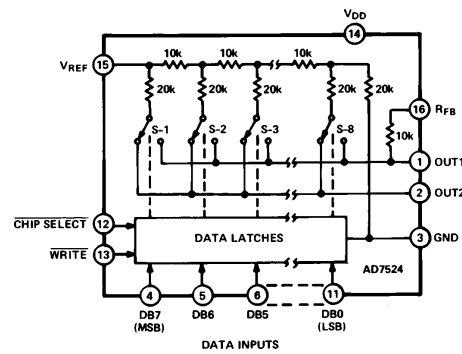
Featuring operation from +5 V to +15 V, the AD7524 interfaces directly to most microprocessor buses or output ports.

Excellent multiplying characteristics (2- or 4-quadrant) make the AD7524 an ideal choice for many microprocessor controlled gain setting and signal control applications.

**REV. B**

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use, nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices.

**FUNCTIONAL BLOCK DIAGRAM**



**ORDERING GUIDE**

Model <sup>1</sup>	Temperature Range	Nonlinearity (V <sub>DD</sub> = +15 V)	Package Option <sup>2</sup>
AD7524JN	-40°C to +85°C	±1/2 LSB	N-16
AD7524KN	-40°C to +85°C	±1/4 LSB	N-16
AD7524LN	-40°C to +85°C	±1/8 LSB	N-16
AD7524JP	-40°C to +85°C	±1/2 LSB	P-20A
AD7524KP	-40°C to +85°C	±1/4 LSB	P-20A
AD7524LP	-40°C to +85°C	±1/8 LSB	P-20A
AD7524JR	-40°C to +85°C	±1/2 LSB	R-16A
AD7524AQ	-40°C to +85°C	±1/2 LSB	Q-16
AD7524BQ	-40°C to +85°C	±1/4 LSB	Q-16
AD7524CQ	-40°C to +85°C	±1/8 LSB	Q-16
AD7524SQ	-55°C to +125°C	±1/2 LSB	Q-16
AD7524TQ	-55°C to +125°C	±1/4 LSB	Q-16
AD7524UQ	-55°C to +125°C	±1/8 LSB	Q-16
AD7524SE	-55°C to +125°C	±1/2 LSB	E-20A
AD7524TE	-55°C to +125°C	±1/4 LSB	E-20A
AD7524UE	-55°C to +125°C	±1/8 LSB	E-20A

**NOTES**

<sup>1</sup>To order MIL-STD-883, Class B processed parts, add/883B to part number.

Contact your local sales office for military data sheet. For U.S. Standard

Military Drawing (SMD) see DESC drawing #5962-87700.

<sup>2</sup>E = Leadless Ceramic Chip Carrier; N = Plastic DIP; P = Plastic Leaded Chip Carrier; Q = Cerdip; R = SOIC.

## AD7524—SPECIFICATIONS ( $V_{REF} = +10\text{ V}$ , $V_{OUT1} = V_{OUT2} = 0\text{ V}$ , unless otherwise noted)

Parameter	Limit, $T_A = +25^\circ\text{C}$		Limit, $T_{MIN}, T_{MAX}^1$		Units	Test Conditions/Comments
	$V_{DD} = +5\text{ V}$	$V_{DD} = +15\text{ V}$	$V_{DD} = 5\text{ V}$	$V_{DD} = +15\text{ V}$		
<b>STATIC PERFORMANCE</b>						
Resolution	8	8	8	8	Bits	
Relative Accuracy						
J, A, S Versions	$\pm 1/2$	$\pm 1/2$	$\pm 1/2$	$\pm 1/2$	LSB max	
K, B, T Versions	$\pm 1/2$	$\pm 1/4$	$\pm 1/2$	$\pm 1/4$	LSB max	
L, C, U Versions	$\pm 1/2$	$\pm 1/8$	$\pm 1/2$	$\pm 1/8$	LSB max	
Monotonicity	Guaranteed	Guaranteed	Guaranteed	Guaranteed		
Gain Error <sup>2</sup>	$\pm 2 1/2$	$\pm 1 1/4$	$\pm 3 1/2$	$\pm 1 1/2$	LSB max	
Average Gain TC <sup>3</sup>	$\pm 40$	$\pm 10$	$\pm 40$	$\pm 10$	ppm/ $^\circ\text{C}$	Gain TC Measured from $+25^\circ\text{C}$ to $T_{MIN}$ or from $+25^\circ\text{C}$ to $T_{MAX}$ $\Delta V_{DD} = \pm 10\%$
DC Supply Rejection, <sup>3</sup> $\Delta\text{Gain}/\Delta V_{DD}$	0.08 0.002	0.02 0.001	0.16 0.01	0.04 0.005	% FSR/% max % FSR/% typ	
Output Leakage Current						
$I_{OUT1}$ (Pin 1)	$\pm 50$	$\pm 50$	$\pm 400$	$\pm 200$	nA max	DB0-DB7 = 0 V; $\overline{\text{WR}}, \overline{\text{CS}} = 0\text{ V}$ ; $V_{REF} = \pm 10\text{ V}$
$I_{OUT2}$ (Pin 2)	$\pm 50$	$\pm 50$	$\pm 400$	$\pm 200$	nA max	DB0-DB7 = $V_{DD}$ ; $\overline{\text{WR}}, \overline{\text{CS}} = 0\text{ V}$ ; $V_{REF} = \pm 10\text{ V}$
<b>DYNAMIC PERFORMANCE</b>						
Output Current Settling Time <sup>3</sup> (to 1/2 LSB)	400	250	500	350	ns max	OUT1 Load = 100 $\Omega$ , $C_{EXT} = 13\text{ pF}$ ; $\overline{\text{WR}}, \overline{\text{CS}} = 0\text{ V}$ ; DB0-DB7 = 0 V to $V_{DD}$ to 0 V.
AC Feedthrough <sup>3</sup>						
at OUT1	0.25	0.25	0.5	0.5	% FSR max	$V_{REF} = \pm 10\text{ V}$ , 100 kHz Sine Wave; DB0-DB7 = 0 V; $\overline{\text{WR}}, \overline{\text{CS}} = 0\text{ V}$
at OUT2	0.25	0.25	0.5	0.5	% FSR max	
<b>REFERENCE INPUT</b>						
$R_{IN}$ (Pin 15 to GND) <sup>4</sup>	5 20	5 20	5 20	5 20	k $\Omega$ min k $\Omega$ max	
<b>ANALOG OUTPUTS</b>						
Output Capacitance <sup>3</sup>						
$C_{OUT1}$ (Pin 1)	120	120	120	120	pF max	DB0-DB7 = $V_{DD}$ ; $\overline{\text{WR}}, \overline{\text{CS}} = 0\text{ V}$
$C_{OUT2}$ (Pin 2)	30	30	30	30	pF max	
$C_{OUT1}$ (Pin 1)	30	30	30	30	pF max	DB0-DB7 = 0 V; $\overline{\text{WR}}, \overline{\text{CS}} = 0\text{ V}$
$C_{OUT2}$ (Pin 2)	120	120	120	120	pF max	
<b>DIGITAL INPUTS</b>						
Input HIGH Voltage Requirement $V_{IH}$	+2.4	+13.5	+2.4	+13.5	V min	
Input LOW Voltage Requirement $V_{IL}$	+0.8	+1.5	+0.5	+1.5	V max	
Input Current $I_{IN}$	$\pm 1$	$\pm 1$	$\pm 10$	$\pm 10$	$\mu\text{A}$ max	$V_{IN} = 0\text{ V}$ or $V_{DD}$
Input Capacitance <sup>3</sup>						
DB0-DB7	5	5	5	5	pF max	$V_{IN} = 0\text{ V}$
$\overline{\text{WR}}, \overline{\text{CS}}$	20	20	20	20	pF max	$V_{IN} = 0\text{ V}$
<b>SWITCHING CHARACTERISTICS</b>						
Chip Select to Write Setup Time <sup>5</sup> $t_{CS}$						See Timing Diagram $t_{WR} = t_{CS}$
AD7524J, K, L, A, B, C	170	100	220	130	ns min	
AD7524S, T, U	170	100	240	150	ns min	
Chip Select to Write Hold Time $t_{CH}$						
All Grades	0	0	0	0	ns min	
Write Pulse Width $t_{WR}$						$t_{CS} \geq t_{WR}$ , $t_{CH} \geq 0$
AD7524J, K, L, A, B, C	170	100	220	130	ns min	
AD7524S, T, U	170	100	240	150	ns min	
Data Setup Time $t_{DS}$						
AD7524J, K, L, A, B, C	135	60	170	80	ns min	
AD7524S, T, U	135	60	170	100	ns min	
Data Hold Time $t_{DH}$						
All Grades	10	10	10	10	ns min	
<b>POWER SUPPLY</b>						
$I_{DD}$	1 100	2 100	2 500	2 500	mA max $\mu\text{A}$ max	All Digital Inputs $V_{IL}$ or $V_{IH}$ All Digital Inputs 0 V or $V_{DD}$

**NOTES**

<sup>1</sup>Temperature ranges as follows: J, K, L versions:  $-40^\circ\text{C}$  to  $+85^\circ\text{C}$   
A, B, C versions:  $-40^\circ\text{C}$  to  $+85^\circ\text{C}$   
S, T, U versions:  $-55^\circ\text{C}$  to  $+125^\circ\text{C}$

<sup>2</sup>Gain error is measured using internal feedback resistor. Full-Scale Range (FSR) =  $V_{REF}$ .

<sup>3</sup>Guaranteed not tested.

<sup>4</sup>DAC thin-film resistor temperature coefficient is approximately  $-300\text{ ppm}/^\circ\text{C}$ .

<sup>5</sup>AC parameter, sample tested @  $+25^\circ\text{C}$  to ensure conformance to specification.

Specifications subject to change without notice.

**AD7524**

**ABSOLUTE MAXIMUM RATINGS\***  
(T<sub>A</sub> = +25°C, unless otherwise noted)

V <sub>DD</sub> to GND	..... -0.3 V, +17 V
V <sub>REFB</sub> to GND	..... ±25 V
V <sub>REF</sub> to GND	..... ±25 V
Digital Input Voltage to GND	..... -0.3 V to V <sub>DD</sub> +0.3 V
OUT1, OUT2 to GND	..... -0.3 V to V <sub>DD</sub> +0.3 V

\*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Power Dissipation (Any Package)	
T <sub>O</sub> +75°C	..... 450 mW
Derates above 75°C by	..... 6 mW/°C
Operating Temperature	
Commercial (J, K, L)	..... -40°C to +85°C
Industrial (A, B, C)	..... -40°C to +85°C
Extended (S, T, U)	..... -55°C to +125°C
Storage Temperature	..... -65°C to +150°C
Lead Temperature (Soldering, 10 secs)	..... +300°C

**CAUTION**

ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although the AD7524 features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.



**TERMINOLOGY**

**RELATIVE ACCURACY:** A measure of the deviation from a straight line through the end points of the DAC transfer function. Normally expressed as a percentage of full scale range. For the AD7524 DAC, this holds true over the entire V<sub>REF</sub> range.

**RESOLUTION:** Value of the LSB. For example, a unipolar converter with n bits has a resolution of (2<sup>-n</sup>) (V<sub>REF</sub>). A bipolar converter of n bits has a resolution of [2<sup>-(n-1)</sup>] [V<sub>REF</sub>]. Resolution in no way implies linearity.

**GAIN ERROR:** Gain Error is a measure of the output error between an ideal DAC and the actual device output. It is measured

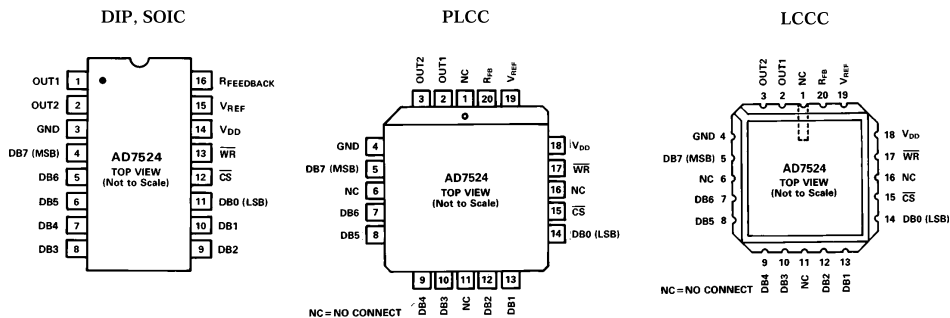
with all 1s in the DAC after offset error has been adjusted out and is expressed in LSBs. Gain Error is adjustable to zero with an external potentiometer.

**FEEDTHROUGH ERROR:** Error caused by capacitive coupling from V<sub>REF</sub> to output with all switches OFF.

**OUTPUT CAPACITANCE:** Capacity from OUT1 and OUT2 terminals to ground.

**OUTPUT LEAKAGE CURRENT:** Current which appears on OUT1 terminal with all digital inputs LOW or on OUT2 terminal when all inputs are HIGH. This is an error current which contributes an offset voltage at the amplifier output.

**PIN CONFIGURATIONS**



## AD7524

CIRCUIT DESCRIPTION  
CIRCUIT INFORMATION

The AD7524, an 8-bit multiplying D/A converter, consists of a highly stable thin film R-2R ladder and eight N-channel current switches on a monolithic chip. Most applications require the addition of only an output operational amplifier and a voltage or current reference.

The simplified D/A circuit is shown in Figure 1. An inverted R-2R ladder structure is used—that is, the binary weighted currents are switched between the OUT1 and OUT2 bus lines, thus maintaining a constant current in each ladder leg independent of the switch state.

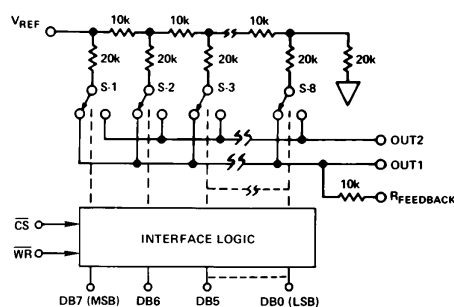


Figure 1. Functional Diagram

## EQUIVALENT CIRCUIT ANALYSIS

The equivalent circuit for all digital inputs LOW is shown in Figures 2. In Figure 2 with all digital inputs LOW, the reference current is switched to OUT2. The current source  $I_{LEAKAGE}$  is composed of surface and junction leakages to the substrate while the  $\frac{1}{256}$  current source represents a constant 1-bit current drain through the termination resistor on the R-2R ladder. The "ON" capacitance of the output N-channel switches is 120 pF, as shown on the OUT2 terminal. The "OFF" switch capacitance is 30 pF, as shown on the OUT1 terminal. Analysis of the circuit for all digital inputs high is similar to Figure 2 however, the "ON" switches are now on terminal OUT1, hence the 120 pF appears at that terminal.

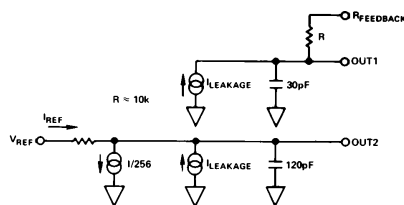


Figure 2. AD7524 DAC Equivalent Circuit—All Digital Inputs Low

INTERFACE LOGIC INFORMATION  
MODE SELECTION

AD7524 mode selection is controlled by the  $\overline{CS}$  and  $\overline{WR}$  inputs.

## WRITE MODE

When  $\overline{CS}$  and  $\overline{WR}$  are both LOW, the AD7524 is in the WRITE mode, and the AD7524 analog output responds to data activity at the DB0–DB7 data bus inputs. In this mode, the AD7524 acts like a nonlatched input D/A converter.

## HOLD MODE

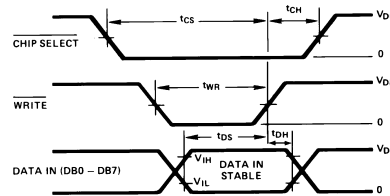
When either  $\overline{CS}$  or  $\overline{WR}$  is HIGH, the AD7524 is in the HOLD mode. The AD7524 analog output holds the value corresponding to the last digital input present at DB0–DB7 prior to  $\overline{WR}$  or  $\overline{CS}$  assuming the HIGH state.

MODE SELECTION TABLE

$\overline{CS}$	$\overline{WR}$	Mode	DAC Response
L	L	Write	DAC responds to data bus (DB0–DB7) inputs.
H	X	Hold	Data bus (DB0–DB7) is Locked Out:
X	H	Hold	DAC holds last data present when $\overline{WR}$ or $\overline{CS}$ assumed HIGH state.

L = Low State, H = High State, X = Don't Care.

## WRITE CYCLE TIMING DIAGRAM



- NOTES:
- All input signal rise and fall times measured from 10% to 90% of  $V_{DD}$ .  $V_{DD} = +5V$ ,  $t_r = t_f = 20ns$ ;  $V_{DD} = +15V$ ,  $t_r = t_f = 40ns$ .
  - Timing Measurement Reference level is  $\frac{V_{IH} + V_{IL}}{2}$ .
  - $t_{DS} + t_{DH}$  is approximately constant at 145ns min at  $+25^\circ C$ ,  $V_{DD} = +5V$  and  $t_{WR} = 170ns$  min. The AD7524 is specified for a minimum  $t_{DS}$  of 10ns, however, in applications where  $t_{DH} > 10ns$ ,  $t_{DS}$  may be reduced accordingly up to the limit  $t_{DS} = 65ns$ ,  $t_{DH} = 80ns$ .

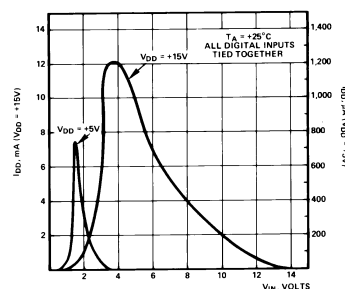


Figure 3. Supply Current vs. Logic Level  
Typical plots of supply current,  $I_{DD}$ , versus logic input voltage,  $V_{IN}$ , for  $V_{DD} = +5V$  and  $V_{DD} = +15V$  are shown above.



AD7524

ANALOG CIRCUIT CONNECTIONS

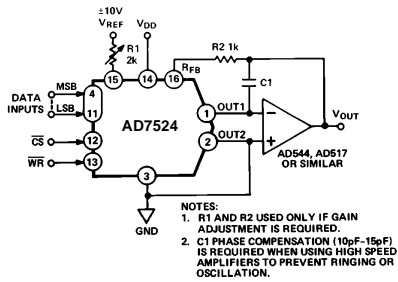


Figure 4. Unipolar Binary Operation (2-Quadrant Multiplication)

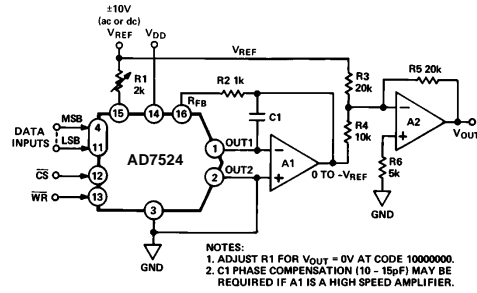


Figure 5. Bipolar (4-Quadrant) Operation

Table I. Unipolar Binary Code Table

Digital Input MSB    LSB	Analog Output
1111 1111	$-V_{REF} (255/256)$
1000 0001	$-V_{REF} (129/256)$
1000 0000	$-V_{REF} (128/256) = -V_{REF}/2$
0111 1111	$-V_{REF} (127/256)$
0000 0001	$-V_{REF} (1/256)$
0000 0000	$-V_{REF} (0/256) = 0$

Note: 1 LSB =  $(2^{-8})(V_{REF}) = 1/256 (V_{REF})$

Table II. Bipolar (Offset Binary) Code Table

Digital Input MSB    LSB	Analog Output
1111 1111	$+V_{REF} (127/128)$
1000 0001	$+V_{REF} (1/128)$
1000 0000	0
0111 1111	$-V_{REF} (1/128)$
0000 0001	$-V_{REF} (127/128)$
0000 0000	$-V_{REF} (128/128)$

Note: 1 LSB =  $(2^{-7})(V_{REF}) = 1/128 (V_{REF})$

MICROPROCESSOR INTERFACE

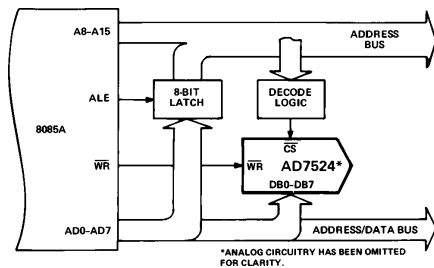


Figure 6. AD7524/8085A Interface

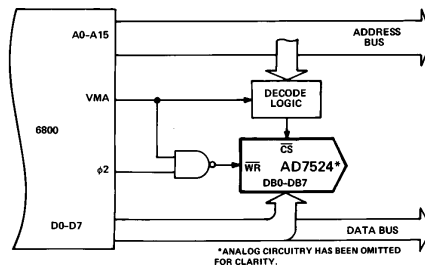


Figure 7. AD7524/MC6800 Interface

AD7524

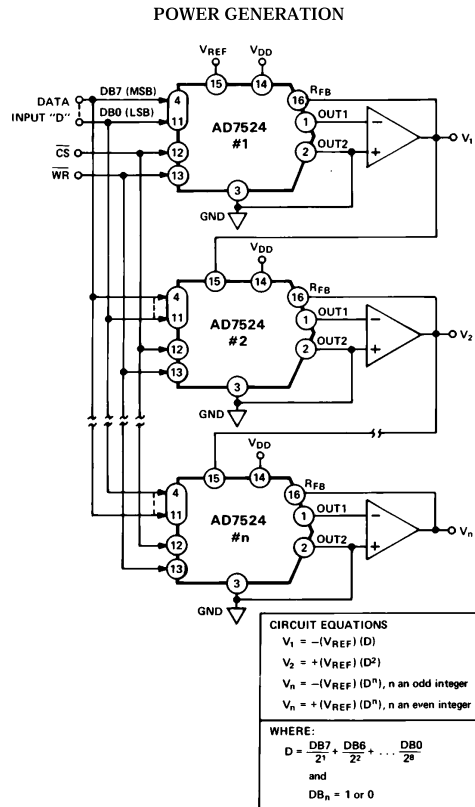
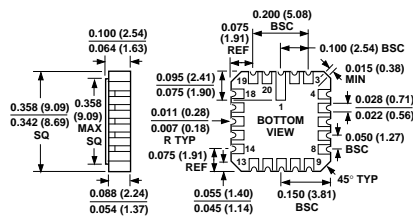


Figure 8.

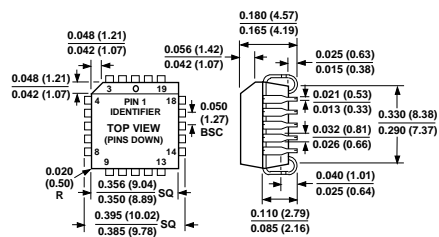
**AD7524**

**OUTLINE DIMENSIONS**  
Dimensions shown in inches and (mm).

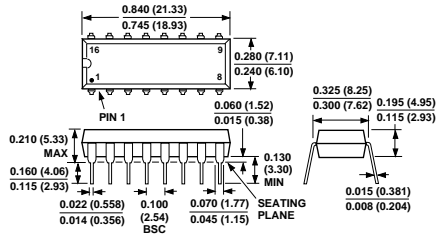
**20-Terminal Ceramic Leadless Chip Carrier (E-20A)**



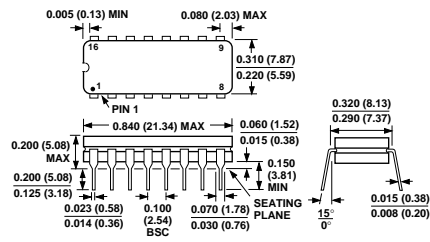
**20-Lead Plastic Leadless Chip Carrier (PLCC) (P-20A)**



**16-Lead Plastic DIP (Narrow) (N-16)**



**16-Lead Cerdip (Q-16)**



**16-Lead Narrow-Body (SOIC) (R-16A)**

