Appendices

HELPFUL HINTS

1) All unused inputs should be tied to a TTL High.

2) Use separate Load and Arm Commands for asynchronous operations.

3) Set (or clear) command should be issued after the load counter command.

4) The counter output should not be connected to edge-sensitive interrupts as the load counter command may cause a glitch on the counter output. Interrupts should be disabled during the load command if this absolutely has to be done.

5) When using fast processors, watch out for recovery time!

6) The Am9513/Am9513A has a peculiarity when using a counter in any mode that alternates the reload source between the load and hold registers (e.g., mode J-variable duty cycle rate generator with no hardware gating). If the counter is disarmed when its value is 0001_H (for down counting) and then armed again, the reload source after the first TC will be the load register instead of the hold register. A software "workaround" is to issue a "dummy" load counter command to the counter immediately after the disarm command.

7) The Am9513A TC output will never go active if programmed to be in the TC toggle mode and the step command is used to increment or decrement the counter. (The Am9513 does not have this problem.)

   The output will go into TC if programmed to be in the active High or active Low terminal count modes. However, the only two ways out of TC are:

   a) Arming the counter and having an active source connected to it, or,

   b) Issuing another step command.

   (The non-A Am9513 counts the next clock pulse, even when not armed, to get out of TC.)

8) For counting rates greater than 7MHz, refer to:

   "High Frequency Operations with the 9513 Controller" by Terrence J. Andrews. Electronics/February 28, 1980.

   "Pulse Swallowing" by John Nichols and Charles Shinn. EDN/October 1, 1970.

APPENDIX A – DEALING WITH METASTABLE PROBLEMS

The Am9513A has improved synchronizing circuitry which limits the error to ±1 count if the specified setup and hold times are not met. This section may be skipped if a ±1 count is acceptable. Please read this section if using the Am9513.

Any circuit employing memory storage devices (i.e., flip-flops) is susceptible to metastable problems. These problems will surface when the storage device is clocked in close temporal proximity to changes on the data inputs. For example, with a D-type flip-flop, changing the D input near an active-going clock edge may cause the flip-flop to go into a metastable state. When a device goes into a metastable state, the Q and $\overline{Q}$ outputs may both be at the same level, may be at intermediate levels or may oscillate between levels. The circuit can remain in the metastable state for a time duration many times longer than the normal propagation delay of the device. The time that the metastable state will persist is based on the exponentially decreasing probability curve (see Figure A-1), with the rate of decrease dependent on the gain-bandwidth product of the device.

Circuit devices such as flip-flops have setup and hold time requirements on the control inputs specified relative to applied clock edges. These setup and hold parameters specify the window during which changes on the control lines may cause a metastable. As long as these parameters are met, metastables will not arise.

Computer system designers eliminate most metastable problems by designing synchronous systems which guarantee the required setup and hold time between clock edges and storage device control inputs. Inputs which are inherently asynchronous, such as interrupts, are sampled and held for a sufficiently long time to allow most metastables to die out. Since the probability of a metastable persisting decreases exponentially, it will never reach zero. Designers of systems with asynchronous inputs use the maximum acceptable failure rate as a guide to determining the time allowed for metastables to decay after sampling the asynchronous input signal.

The data sheet parameters pertaining to synchronization for the Am9513 are TGVEH, TEHGV, TEHWH, TWHEH, TGVWH and TWHGV and for the Am8253 are t_{GS} and t_{GH}. As long as these parameters are met, there will be no metastable problems. To determine if a circuit meets these parameters, use the following guidelines.

Parameters TGVEH and TEHGV for the Am9513 and t_{GS} and t_{GH} for the 8253 specify that when a source edge is applied to an armed counter, the gate must be stable TGVEH (Am9513) or t_{GS} (8253) before the source edge and the gate must be held stable for TEHGV (Am9513) or t_{GH} (8253) after the source edge for level gating; and for edge gating meaningful transitions on the gate input must not occur closer than TGVEH (Am9513) or t_{GS} (8253) before the source edge or sooner than TEHGV (Am9513) or t_{GH} (8253) after the source edge. A meaningful transition in edge gating is defined as a gate edge applied to an armed counter that starts the counter counting in Modes C, F, I, L, O or R, (Am9513) or any active-going gate edge applied to an armed counter in Modes O or R. In the 8253, parameters t_{GS} and t_{GH} must be met relative to the falling source edge; in the Am9513, parameters TGVEH and TEHGV should be met relative to the rising source edge if Counter Mode register bit CM12 = 0 and to the falling source edge if CM12 = 1. These two parameters need only be met while the counter is armed.

Parameters TEHWH and TWHEH (Am9513) specify that when a write command is issued to a given counter, that counter must not have an active-going count source edge any closer than TEHWH before the rising $\overline{WR}$ edge and the next active-going count source edge must not occur until TWHEH after the rising $\overline{WR}$ edge. These parameters must be met for all commands issued to an armed counter and for ARM, LOAD-and-ARM and STEP commands issued to a disarmed counter. As an aside, if these parameters can not be met, it is strongly suggested that the programmer use separate LOAD and ARM commands rather than the combined LOAD-and-ARM command in order to limit the number of events that need to be simultaneously performed. The 8253 has a similar restriction, namely that applying a rising $\overline{WR}$ edge to a counter close to our active-going source edge may cause counter problems. However, the 8253's data sheet, unlike the Am9513's, does not call out what constitutes a safe application of a write to counter.

Parameters TGVWH and TWHGV (Am9513) specify that when a write command is issued to a given counter, the gate input to that counter must be stable TGVWH before the rising $\overline{WR}$ edge and the gate must remain stable TWHGV after the rising $\overline{WR}$ edge for level gating; and for edge gating meaningful transitions (defined above) on the gate must not occur TGVWH before the rising $\overline{WR}$ edge or sooner than TWHEH after the rising $\overline{WR}$ edge. These parameters must be met for all commands issued to an armed counter and for ARM, LOAD-and-ARM and STEP commands issued to a disarmed counter. As above, use the separate LOAD and ARM commands if these parameters can not be met. The 8253 counters are susceptible to erroneous operation if a rising $\overline{WR}$ edge is applied to a counter close to

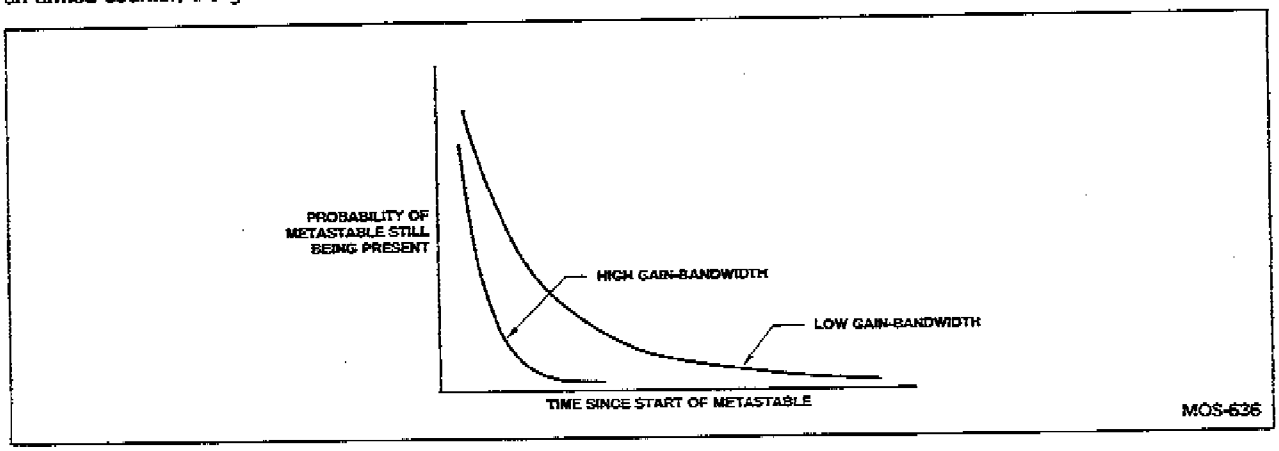


Figure A-1. Metastable Decay Probabilities

a retriggering gate edge. Unlike the Am9513's data sheet, the 8253's does not specify when it is safe to apply a write command.

In the Am9513, some commands such as ARM, DISARM and SAVE can be applied to more than one counter at a time. For such commands, each counter being acted on must meet parameters TEHWH, TWHEH, TGVWH and TWHGV.

Failure to meet one or more of the Am9513 parameters TEHWH, TNHEH, TGVWH or TWHGV or failure to meet the unspecified 8253 restrictions or when write commands can be issued may result in incorrect execution of the entered command. For example, with a Am9513 SAVE command or an 8253 counter latching operation, an incorrect value might be stored. It is also possible, although likely less probable, that a command entered in violation of these parameters may have a more drastic effect, perhaps altering the counter's contents or perhaps locking the counter up. It is because of the unpredictable nature of metastables that it is not possible to pinpoint the command failure mode that may be experienced. Many users provide reasonableness checking routines in their software to help detect if an error arises, perhaps performing two (Am9513) SAVE or two (8253) counter latching operations, for example, and comparing the results. The Am9513A may have a count error of ±1.

With the Am9513, it is sometimes necessary to calculate the relationship between one of the internal F signals (F1 through F5) and some externally applied signal, perhaps a gate or $\overline{WR}$ edge. If FOUT is driven from an internal F signal, say F5, any internal F signal, say F3, can be referred to the FOUT transition using data sheet parameters TFN and TEHFV. In the above example, the maximum delay between a transition on F3 and a transition on FOUT is given by TFN (F3 to F4 delay) + TFN (F4 to F5 delay) + TEHFV (FOUT source to FOUT output delay) = 2 TFN + TEHFV. It turns out that the internal F1 signal can be assumed to be equivalent to (i.e., has 0 ns skew relative to) the X2 input. This can be used to reference X2 to one of the internal F signals, for example, to F3. The X2 to F3 maximum delay is simply: (0 ns (X2 to F1 delay) + TFN (F1 to F2 delay) + TFN (F2 to F3 delay)) = 2 TFN. Using the above techniques, any F signal can be referenced to FOUT (if FOUT has one of the F signals as a source) or to X2. chronize signals, perhaps an asynchronous gate or $\overline{WR}$ edge, relative to the internal F signal. Note, however, to avoid biasing Since the relationship between X2 or FOUT to the internal F signal is known, the X2 or FOUT signal may be used to synproblems with the Am9513's internal oscillator, if X2 connects to a crystal or to an LC or RC network, the X2 node should not be used to drive any gates other than the Am9513's X2 input. The X2 signal should only be used to drive other gates if it is being driven, as shown in Figure 2-3 (c), by an external signal not generated by the Am9513's internal oscillator.

In many systems it may be difficult or impossible to meet some or all of these parameters. In recognition of this, the Am9513 has on-board circuitry designed to synchronize signals violating the above parameters. Because of the low gain-bandwidth product available in MOS technology vis-a-vis bipolar technology, there is a small but significant probability of failure in the Am9513's synchronizing circuitry. Again at the risk of being repetitious, problems because of synchronization failure can only occur if the user does not meet data sheet parameters TGVEH, TEHGV, TEHWH, TWHEH, TGVWH and TWHGV (Am9513) or t_{GS} and t_{GH} (8253).

Calculations can be made to compute the expected failure rate for truly asynchronous signal entry. To calculate the expected metastable error rate, it must be assumed that the two input signals being analyzed are truly asynchronous. Signals which are generated from a common source but which may be skewed relative to each other because of indeterminate propagation delays in their separate paths are not truly asynchronous. The word synchronous, as used in this document, refers to all of these non-asynchronous signals, i.e., to any signals with a predictable timing relationship.

The reason the signal must be truly asynchronous is related to the setup/hold window specified by the parameters in the data sheet. Any given chip will have a setup/hold window much smaller than that given in the data sheet. The typical setup/hold window requirements for a particular counter may actually be many orders of magnitude smaller than the one given in the data sheet. See Figure A-2. This very narrow window for different parts will appear at different points within the data sheet's required setup/hold window. Also, for a given part, this very narrow window will move with changes in operating conditions (voltage, temperature, etc.). The data sheet parameters specify the outer limits of these variations for worst-case conditions.

If signal transitions occur within the narrow setup/hold window of a given part, there is a high probability of a metastable occurring. If the applied signals violating the setup/hold time are truly asynchronous, the probability of their violating the setup/hold window is random. If, on the other hand, the two signals are derived from a common signal or crystal, they in fact will have some predictable

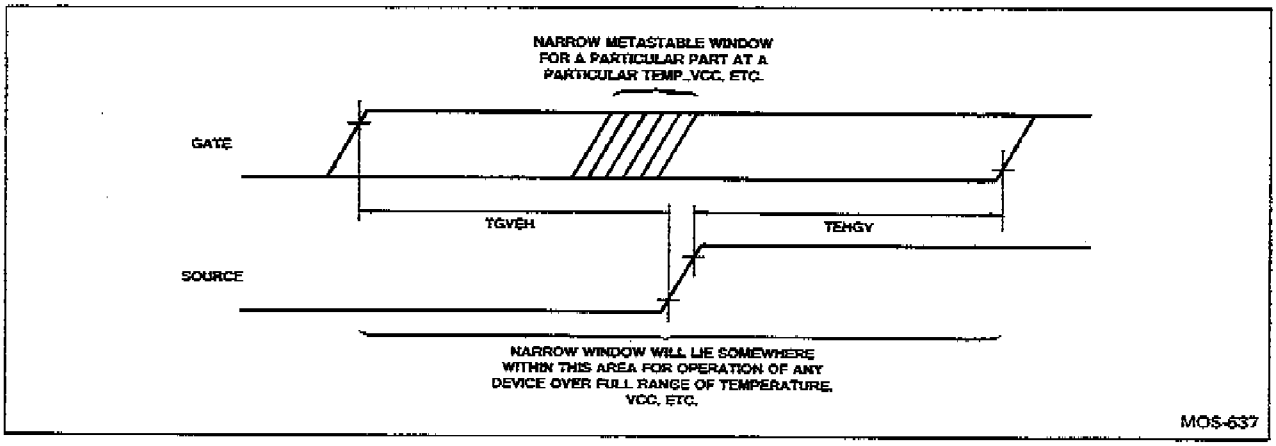


Figure A-2. Am9513 Setup-Hold Requirements

APPENDIX A (Cont.)

relationship and, at a given set of operating conditions, may fall in the setup/hold window time after time, causing high failure rates.

For this reason, if the Am9513's or 8253's synchronization parameters can not be met, the violating signal pairs must be truly asynchronous to each other in order to set the probability odds on the side of the user. More specifically, if parameters TGVEH or TEHGV (Am9513) or t_{GS} or t_{GH} (8253) can not be met, the gate and source should be asynchronous. If TEHWH or TWHEH (Am9513) can not be met, the $\overline{WR}$ and source signals should be asynchronous (i.e., do not clock the Am9513 off the processor's clock, or vice versa). If TGVWH or TWHGV can not be met, the $\overline{WR}$ and gate signals should be asynchronous. Since the 8253's data sheet does not specify when it is safe to issue write commands the user cannot be certain metastables will not occur.

Assume there is a circuit which violates at least one of the Am9513's or 8253's synchronization parameters and an approximation is desired of the failure rate that may occur. To a first order approximation, one may use the formula

$$P(meta) = r_1\, r_2\, t_w$$

where

P(meta) = probability of a metastable over a given period of time

r_1, r_2 = transition rates of two input signals

t_w = setup/hold window size of a particular part

Note that the equations presented in this appendix are useful for analyzing the probability of metastables, not only for the Am9513 and 8253 but also for any other sequential logic circuit having asynchronous inputs.

Variables r_1 and r_2 are the rate of relevant transitions on the two input signals. For Am9513 sources, only the active-going source edge must be synchronized to the gate or command input. The inactive-going source edge has no effect on the counter. For the 8253, the falling source edge must be synchronized to the gate. For Am9513 command entry, only the rising (trailing) $\overline{WR}$ edge must be synchronized; the falling (leading) $\overline{WR}$ edge can be asynchronous. In Am9513 level gating modes and 8253 modes 0, 2, 3 and 4, both active and inactive gate edges must be synchronized to the source, whereas in Am9513 edge gating applications or 8253 modes 1 and 5, only the meaningful gate edges have significance. (See the definition of meaningful gate edges earlier in this appendix.) Thus for all inputs except Am9513 level gate signals or 8253 gate signals in modes 0, 2, 3 and 4, parameters r_1 or r_2 are equal to the frequencies of the inputs. For level gating, since both edges have significance, the transition rate r_1 or r_2 is twice the gate signal frequency.

Variable t_w is the narrow setup/hold window for a given part under a certain set of operating conditions (see Figure A-2). The actual value of this variable will be highly dependent on the application, and for this reason it is best determined empirically for the particular application being studied.

One approach useful in determining P(meta) is to run the application at a much higher rate than is normally intended. This should generate an easily measurable error rate. The value of t_w can then be calculated and used to calculate P(meta) for the normal rate of operation.

For example, suppose a user plans to asynchronously level-gate at a rate of 10Hz while using a 1000Hz source. The user wishes to calculate P(meta) for this application. Parameter t_w might be calculated while running a 2MHz signal into the source while level-gating at 200kHz. He might observe the counter output to verify correct operation. By observing the error rate over a period of time, the above formula could be used to calculate t_w, as follows:

$$t_w = \frac{P(meta)'}{r_1'\, r_2'}$$

Note in our application $r_1' = 2 \times 10^6$ and $r_2' = 400 \times 10^3$. Parameter r_2' is twice the gate frequency since both gate edges must be synchronized. The empirically determined t_w can now be used to find P(meta) for the intended application using the formula

$$P(meta) = r_1\, r_2\, t_w$$

with r_1 = 1000Hz and r_2 = 20Hz, corresponding to the source and gate repetition rates for the desired application.

In many cases, no errors will be observable even at accelerated operating rates. The ratio of the accelerated rate to the final applications rate allows extrapolation of the period of time likely between errors. For example, if our user observed the accelerated application for ten minutes and no error was seen, he could extrapolate the time between errors for the final application as follows:

$$P(meta) = \frac{r_1\, r_2\, P(meta)'}{r_1'\, r_2'} = \frac{1000\ Hz \times 20\ Hz \times \frac{1\ error}{10\ minutes}}{2 \times 10^6\ Hz \times 400 \times 10^3\ Hz}$$

$$= 2.5 \times 10^{-9}\ errors/sec = 1\ error/4 \times 10^8\ minutes$$

Thus, the final application's average error rate will likely be better than once every 760 years.

In some applications, more than one Am9513 setup/hold parameter may be violated. (Since the 8253 data sheet only specifies the source-gate setup requirements, the user can not be sure whether applied write commands are in violation of the setup and hold requirements of the 8253's gate and source.) For each setup/hold window possibly violated (TGVEH and TEHGV; TEHWH and TWHEH; and TGVWH and TWHGV), the related pairs of signals (gate-source; source – $\overline{WR}$; and gate – $\overline{WR}$) should be run at accelerated rates to compute t_w'.

The formula can then be used to estimate P(meta) for each of the setup/hold requirements. The resultant error probability is then the sum of the P(meta) values for each setup/hold window.

For example, our above discussion calculated P(meta) for the TGVEH – TEHGV source-gate setup/hold window. We might also calculate P(meta) for the TEHWH – TWHEH source – $\overline{WR}$ window by using accelerated source and WR signals, keeping the gate repetition rate at the final application rate of 10Hz to minimize its effect. If our new source – $\overline{WR}$ P(meta) was 1 error every 500 years, the combined average error rate would be (1/500 + 1/760) or 1 error every 300 years. If the gate – $\overline{WR}$ setup/hold parameters were also violated, the P(meta) for this window should also be calculated as a component in the final error rate. As mentioned earlier, these calculations can also be used to analyze metastable error rates for any other sequential logic circuit having asynchronous inputs.

Three cautionary notes are appropriate at this point. First, the formula provides a first order approximation only and should be used only for rough estimates. Second, since metastable errors will occur at random times for asynchronous signals, multiple measurements and use of statistical techniques should be used to minimize the effect of random fluctuations in the measured data. Third, the desired application should be thoroughly tested at the normal operating frequencies. This approximation technique of estimating error rates should be used to further substantiate the results of this through testing rather than be used in lieu of thoroughly testing the application.

APPENDIX B – KEY TO TIMING DIAGRAMS

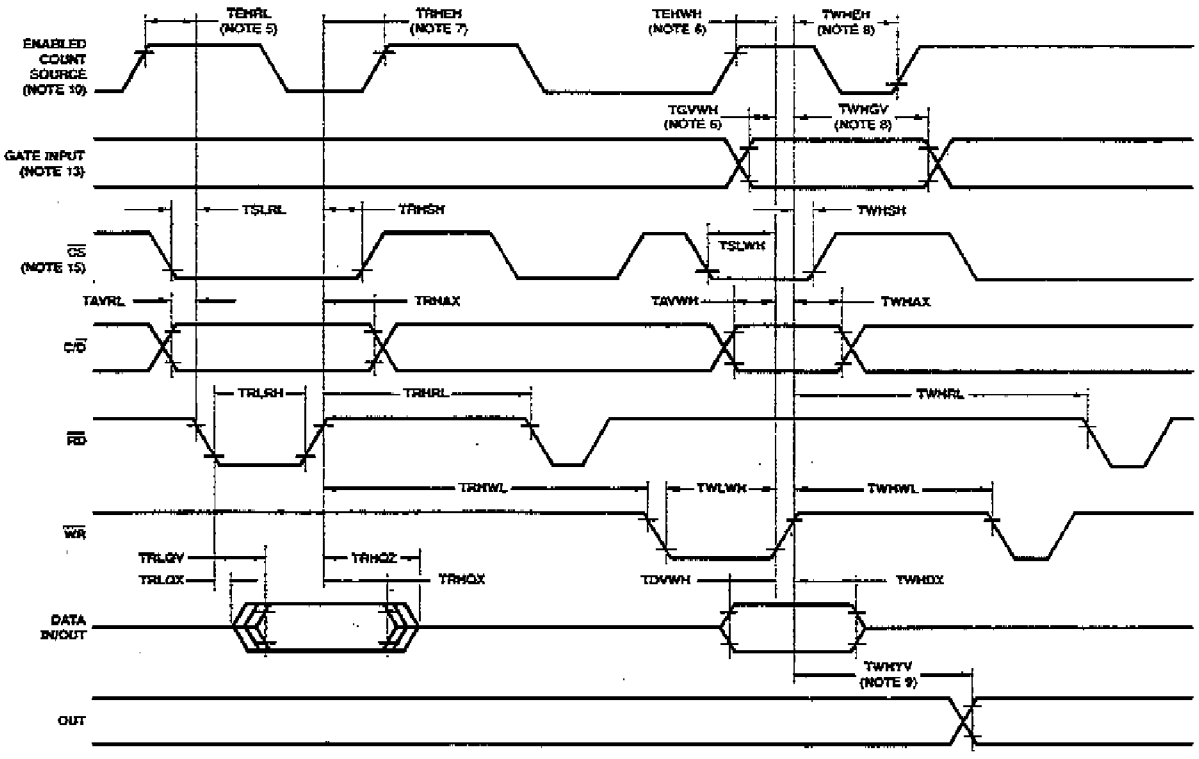
Waveform Representations

Waveform	Inputs	Outputs
	Must be steady	Will be steady
	May change from High to Low	Will be changing from High to Low
	May change from Low to High	Will be changing from Low to High
	Don't care Any change permitted	Changing State unknown
	May make one change from High to Low or Low to High	Will make one change from High to Low or Low to High

MOS-638

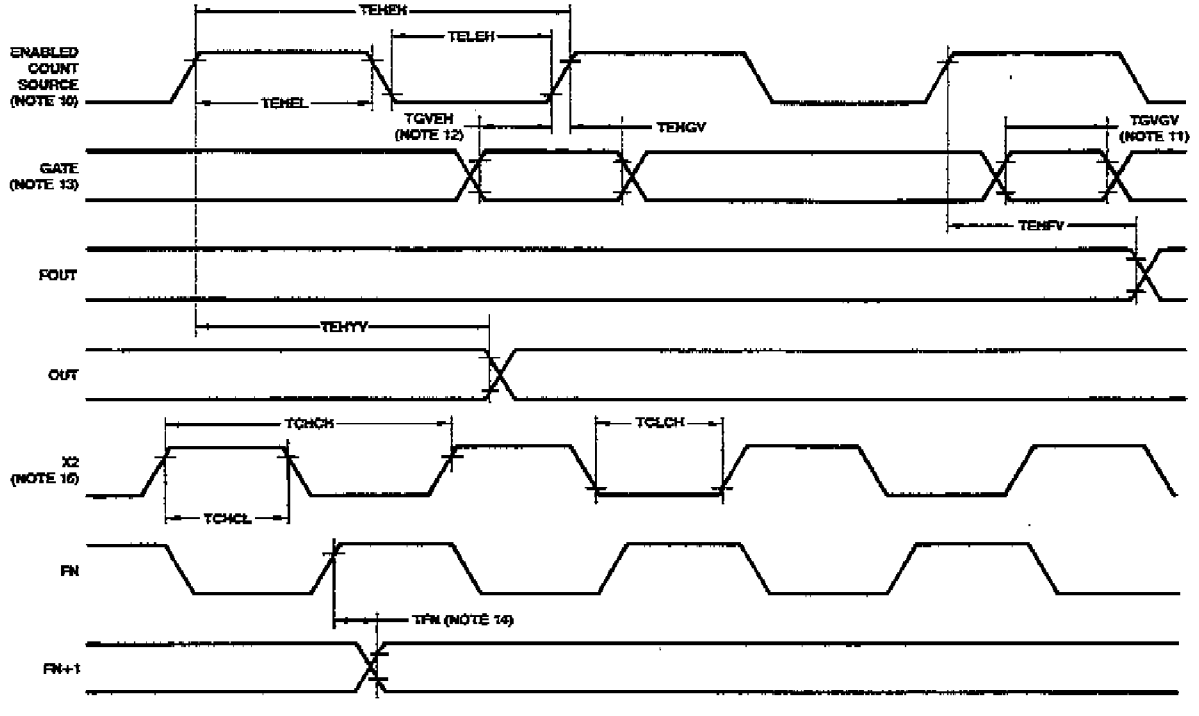
APPENDIX B – Am9513 SWITCHING WAVEFORMS

Bus Transfer Switching Waveforms



MOS-183

Counter Switching Waveforms



MOS-184

APPENDIX B – Am9513 SWITCHING WAVEFORMS

NOTES:

1. Typical values are for T_A = 25°C, nominal supply voltage and nominal processing parameters.
2. Test conditions assume transition times of 10ns or less, timing reference levels of 0.8V and 2.0V and output loading of one TTL gate plus 100pF, unless otherwise noted.
3. Abbreviations used for the switching parameter symbols are given as the letter T followed by four or five characters. The first and third characters represent the signal names on which the measurements start and end. Signal abbreviations used are:

   A (Address) = C/$\overline{D}$
   C (Clock) = X2
   D (Data In) = DB0-DB15
   E (Enabled counter source input) = SRC1-SRC5, GATE1-GATE5, F1-F5, TCN–1
   F = FOUT
   G (Counter gate input) = GATE1-GATE5, TCN–1
   Q (Data Out) = DB0-DB15
   R (Read) = $\overline{RD}$
   S (Chip Select) = $\overline{CS}$
   W (Write) = $\overline{WR}$
   Y (Output) = OUT1-OUT5

   The second and fourth letters designate the reference states of the signals named in the first and third letters respectively, using the following abbreviations.

   H = High
   L = Low
   V = Valid
   X = unknown or don't care
   Z = high impedance

4. Switching parameters are listed in alphabetical order.
5. Any input transition that occurs before this minimum setup requirement will be reflected in the contents read from the status register.
6. Any input transition that occurs before this minimum setup requirement will act on the counter before the execution of the operation initiated by the write. Failure to meet this setup time when issuing commands to the counter may result in incorrect counter operations for the Am9513. For the Am9513A, the count value may be off by ±1. The counter will continue to operate correctly.
7. Any input transition that occurs after this minimum hold time is guaranteed to not influence the contents read from the status register on the current read operation.
8. Any input transition that occurs after this minimum hold time is guaranteed to be seen by the counter as occurring after the action initiated by the write operation. Failure to meet this hold time when issuing commands to the counter may result in incorrect counter operation for the Am9513. For the Am9513A, the count value may be off by ±1. The counter will continue to operate correctly.
9. This parameter applies to cases where the write operation causes a change in the output bit.
10. The enabled count source is one of F1-F5, TCN–1, SRC1-SRC5 or GATE1-GATE5, as selected in the applicable Counter Mode register. The timing diagram assumes the counter counts on rising source edges. The timing specifications are the same for falling-edge counting.
11. This parameter applies to edge gating (CM15-CM13 = 110 or 111) and gating when both CM7 = 1 and CM15-CM13 ≠ 000. This parameter represents the minimum GATE pulse width needed to ensure that the pulse initiates counting or counter reloading.
12. This parameter applies to both edge and level gating (CM15-CM13 = 001 through 111) and gating when both CM7 = 1 and CM15-CM13 = 000. This parameter represents the minimum setup or hold times to ensure that the Gate input is seen at the intended level on the active source edge. Failure to meet the required setup and hold times may result in incorrect counter operation for the Am9513. For the Am9513A, the count value may be off by ±1. The counter will continue to operate correctly.
13. This parameter assumes that the GATENA input is unused (16-bit bus mode) or is tied high. In cases where the GATENA input is used, this timing specification must be met by both the GATE and GATENA inputs.
14. Signals F1-F5 cannot be directly monitored by the user. The phase difference between these signals will manifest itself by causing counters using two different F signals to count at different times on nominally simultaneous transitions in the F signals.
15. This timing specification assumes that $\overline{CS}$ is active whenever $\overline{RD}$ or $\overline{WR}$ are active. $\overline{CS}$ may be held active indefinitely.
16. This parameter assumes X2 is driven from an external gate with a square wave.
17. This parameter assumes that the write operation is to the command register.

The following Am9513 data model summary collects together all C structures appearing in the text of this note for reference purposes.

```
/*
        Record definitions - register fields for Am9513 data model
*/
                                /* Master mode Register */

typedef RECORD  {unsigned day_mode      : 2 ;
                 unsigned compar_1      : 1 ;
                 unsigned compar_2      : 1 ;
                 unsigned FOUT_source   : 4 ;
                 unsigned FOUT_divisor  : 4 ;
                 unsigned FOUT_gate     : 1 ;
                 unsigned data_bus      : 1 ;
                 unsigned data_ptr      : 1 ;
                 unsigned scaler        : 1 ;
                } master_type ;

                                /* Counter Mode Register */

typedef RECORD  {unsigned output        : 3 ;
                 unsigned direction     : 1 ;
                 unsigned base          : 1 ;
                 unsigned control       : 3 ;
                 unsigned source        : 4 ;
                 unsigned edge          : 1 ;
                 unsigned gate          : 3 ;
                } count_type ;

                                /* Status Register      */

typedef RECORD  {unsigned byte_ptr      : 1 ;
                 unsigned output1       : 1 ;
                 unsigned output2       : 1 ;
                 unsigned output3       : 1 ;
                 unsigned output4       : 1 ;
                 unsigned output5       : 1 ;
                 unsigned not_used      : 2 ;
                } status_type ;

                                /* Load data ptr register command */

typedef RECORD  {unsigned group         : 3 ;
                 unsigned element       : 2 ;
                 unsigned cmnd_code     : 3 ;
                } data_type ;

/*      - chip level structures */

                                /* Single counter set   */

typedef RECORD  {count_type     mode ;
                 unsigned int   load ;
                 unsigned int   hold ;
                } channel_type ;

                                /* Am9513 chip set      */

typedef RECORD  {master_type    master ;
                 channel_type   counter [5] ;
                 status_type    status ;
                 unsigned int   alarm_1 ;
                 unsigned int   alarm_2 ;
                } AM9513_type ;
```

Appendix D – Am9513 Macro Command Summary

This appendix defines the macro command available with detailed syntax of use. The macros are available for the following assemblers:

1. Am8080/Am8085 MACR08 assembler (AMD)
2. Z80 RIO assembler (Zilog)
3. Z8000 MACZ assembler (AMD)

The available macros are summarised here with details of usage. Notice that in the following text angle brackets ("<>") are used to describe parameters to be entered and as such are not actually to be entered. Square brackets ("[]") describe optional items.

The "OR" symbol ("¦") signifies that only one of the parameters given should be entered.

The comma is a legal parameter separator for all three macro files.

RESET	
FOUT	ON¦OF
DPS	ON¦OF
POINT	<group>,<element>
ARM	<combination>
LOAD	<combination>
LD_ARM	<combination>
SAVE	<combination>
DRMSAV	<combination>
SET_	<counter>
CLEAR	<counter>
STEP	<counter>
LOAD_REG	<counter>,<constant>[,<indirection>]
HOLD_REG	<counter>,<constant>[,<indirection>]
MODE_REG	<counter>,<output>,<direction>,<base>,<control>,<source>,<edge>,<gate>
MASTER	<day_mode>,<compar_1>,<compar_2>,<FOUT_source>,<FOUT_divisor>,<FOUT_gate>,<data_bus>,<data_ptr>,<scaler>

```
<combination>    ::=    <counter>|<counter>,<combination>

<counter>        ::=    1|2|3|4|5

<indirection>    ::=    I

<group>          ::=    <counter>|CTRL_GR

<element>        ::=    MODE_|LOAD_|HOLD_|HOLD_CY
                        |ALARM1_|ALARM2_|MASTER_|STATUS_

<output>         ::=    OF_LO_TC|ACT_HI_TC|TC_TOGGLE|OF OC_TC|ACT_LO_TC

<direction>      ::=    UP|DOWN

<base>           ::=    <scaler>

<control>        ::=    <allowed_modes>

<source>         ::=    <FOUT_source>|TC_NM1

<edge>           ::=    RISE|FALL

<gate>           ::=    NO_GATE|HL_TC_NM1|HL_NP1_GATE|HL_NM1_GATE|HL_GATE_N
                        |LL_GATE_N|HE_GATE_N|LE_GATE_N

<day_mode>       ::=    TOD_OFF|TOD_50HZ|TOD_60HZ|TOD_100HZ


<compar_2>       ::=    <compar_1>

<compar_1>       ::=    ENABLE|DISABLE

<FOUT_source>    ::=    SRC_1|SRC_2|SRC_3|SRC_4|SRC_5
                        |GATE_1|GATE_2|GATE_3|GATE_4|GATE_5
                        |F1|F2|F3|F4|F5

<FOUT_divisor>   ::=    <hex_digit>

<FOUT_gate>      ::=    ON|OF

<data_bus>       ::=    BUS_8|BUS_16

<data_ptr>       ::=    ON|OF

<scaler>         ::=    BINARY|BCD

<hex_digit>      ::=    <digit>|0AH|0BH|0CH|0DH|0EH|0FH

<constant>       ::=    <digit>|<digit><constant>

<digit>          ::=    0|1|2|3|4|5|6|7|8|9

<allowed_modes>  ::=    MODE_ABC|MODE_DEF|MODE_GHI|MODE_JKL|MODE_mNO
                        |MODE_pQR|MODE_Stu|MODE_Vwx
```

The following macro code definitions implement the macro commands as listed in the Macro Command Summary (Appendix C). Notice that all macros use only the A and F registers, all other working registers are unaffected. These macros are targeted for the AMD MACRO8 assembler.

```
;       Am9513 Macro definitions for Am8080 Macro8 Assembler


S_MASK  MACRO   P1,P2,P3,P4,P5

        ;; Recursive macro to get correct s-field

        IFNB    <P2>
        S_MASK  P2,P3,P4,P5
        ENDIF
        IFT     (0 LT P1) AND (P1 LT 6)
DLAB    SET     DLAB OR (1 SHI (P1-1))
        ELSE
        ??? P1  ; Illegal Counter
        ENDIF
        ENDM

S_TYPE  MACRO   X,P1,P2,P3,P4,P5

        ;; Optimised sequence for multiple register s-field commands

        IFT     X EQ 20H OR X EQ 40H OR X LQ 60H OR X EC 2C0H OR X EQ 0A0H
DLAB    SET     X                                                  OR X EQ 80H
        S_MASK  P1,P2,P3,P4,P5
        MVI     A,DLAB
        OUT     A_CTRI
        ELSE
        ??? X   ; Illegal s-mask command code
        ENDIF
        ENDM

N_TYPE  MACRO   P1,P2

        ;; Optimised sequence for single register n-field commands

        IFT     (0 LT P1) AND (P1 LT 6)
        MVI     A,P1 OR P2
        OUT     A_CTRL
        ELSE
        ??? P1  ; Illegal Counter
        ENDIF
        ENDM

ARM     MACRO   P1,P2,P3,P4,P5          ;; Arm counters, any from 1,5
        S_TYPE  20H,P1,P2,P3,P4,P5
        ENDM

LOAD    MACRO   P1,P2,P3,P4,P5          ;; Load counters, any from 1,5
        S_TYPE  40H,P1,P2,P3,P4,P5
        ENDM

LD_ARM  MACRO   P1,P2,P3,P4,P5          ;; Load 'n Arm counters, any from 1,5
        S_TYPE  60H,P1,P2,P3,P4,P5
        ENDM
```

```
DISARM  MACRO   P1,P2,P3,P4,P5          ;; Disarm counters, any from 1,5
        S_TYPE  0C0H,P1,P2,P3,P4,P5
        ENDM

SAVE    MACRO   P1,P2,P3,P4,P5          ;; Save counters, any from 1,5
        S_TYPE  0A0H,P1,P2,P3,P4,P5
        ENDM

DRMSAV  MACRO   P1,P2,P3,P4,P5          ;; Disarm 'n Save counters. any from 1,5
        S_TYPE  80H,P1,P2,P3,P4,P5
        ENDM

SET_    MACRO   P1                      ;; Set single counter output
        N_TYPE  P1,0E8H
        ENDM

CLEAR   MACRO   P1                      ;; Clear single counter output
        N_TYPE  P1,0L0H
        ENDM

STEP    MACRO   P1                      ;; Step single counter output
        N_TYPE  P1,0F0H
        ENDM

FOUT    MACRO   P1                      ;; Gate FOUT on or off
        IFT     '&P1' EQ 'OF'
        MVI     A,0EEH
        OUT     A_CTRL
        ELSE
        IFT     '&P1' EQ 'ON'
        MVI     A,0E6H
        OUT     A_CTRL
        ELSE
        ??? P1  ; Illegal request
        ENDIF
        ENDIF
        ENDM

RESET   MACRO                           ;; Reset, load all counters
        MVI     A,0FFH
        OUT     A_CTRL
        LOAD    1,2,3,4,5
        ENDM

POINT   MACRO   P1,P2           ;; Set data pointer register group, element

        IFT     ((0 LT P1 AND P1 LT 6) OR P1 EQ 7) AND 0 LE P2 AND P2 LT 4
        MVI     A,P1 OR (P2 SHL 3)
        OUT     A_CTRL
        ELSE
        ??? P1,P2       ; Illegal request
        ENDIF
        ENDM
```

```
DPS     MACRO   P1              ;; Set data pointer sequencing on/of
        IFT     '&P1' EQ 'ON'
        MVI     A,0E0H
        OUT     A_CTRL
        ELSE
        IFT     '&P1' EQ 'OF'
        MVI     A,0E8H
        OUT     A_CTRL
        ELSE
        ??? P1  ; Illegal request
        ENDIF
        ENDIF
        ENDM


MASTER  MACRO   P1,P2,P3,P4,P5,P6,P7,P8,P9
                                        ;; Set Master register
DLAB    SET     P1                      ;;  .day_mode
DLAB    SET     DLAB OR (P2 SHL 2)      ;;  .compar_1
DLAB    SET     DLAB OR (P3 SHL 3)      ;;  .compar_2
DLAB    SET     DLAB OR (P4 SHL 4)      ;;  .FOUT_source
DLAB    SET     DLAB OR (P5 SHL 8)      ;;  .FOUT_divisor
DLAB    SET     DLAB OR (P6 SHL 12)     ;;  .FOUT_gate
DLAB    SET     DLAB OR (0 SHL 13)      ;;  .data_bus - 8 bit only
DLAB    SET     DLAB OR (P8 SHL 14)     ;;  .data_ptr
DLAB    SET     DLAB OR (P9 SHL 15)     ;;  .scaler
        POINT   CTRL_GR,MASTER_         ;; Point to Master Register
        MVI     A, LOW DLAB
        OUT     A_DATA                  ;; Send Low byte first
        MVI     A, HIGH DLAB
        OUT     A_DATA                  ;; Then High byte
        ENDM

MODE_REG MACRO  P0,P1,P2,P3,P4,P5,P6,P7
                                        ;; Set counter P0 Mode register
        IFT     P1 EQ 3 OR 5 LT P1
        ??? P1  ; Illegal Output Control
        ELSE
DLAB    SET     P1                      ;;  .output
DLAB    SET     DLAB OR (P2 SHL 3)      ;;  .direction
DLAB    SET     DLAB OR (P3 SHL 4)      ;;  .base
DLAB    SET     DLAB OR (P4 SHL 5)      ;;  .control
DLAB    SET     DLAB OR (P5 SHL 8)      ;;  .source
DLAB    SET     DLAB OR (P6 SHL 12)     ;;  .edge
DLAB    SET     DLAB OR (P7 SHL 13)     ;;  .gate
        IFT     0 LT P0 AND P0 LT 6
        POINT   P0,MODE_                ;; Point to counter P1 Mode
        MVI     A, LOW DLAB
        OUT     A_DATA                  ;; Send Low byte
        MVI     A, HIGH DLAB
        OUT     A_DATA                  ;; Then High byte
        ELSE
        ??? P0  ; Illegal counter #
        ENDIF
        ENDIF
        ENDM
```

```
LOAD_REG MACRO  P0,P1,P2        ;; Set Load reg of counter P0 to P1
         IFNB   <P2>            ;; Test for indirection
         POINT  P0,LOAD_        ;; Nested macro must lie within the test
         LDA    P1
         OUT    A_DATA          ;; Low byte
         LDA    P1+1
         OUT    A_DATA          ;; High byte
         ELSE
         POINT  P0,LOAD_
         MVI    A, LOW P1
         OUT    A_DATA          ;; Low byte
         MVI    A, HIGH P1
         OUT    A_DATA          ;; High byte
         ENDIF
         ENDM

HOLD_REG MACRO  P0,P1,P2        ;; Set Hold Reg of counter P0 to P1
         IFNB   <P2>            ;; Test for indirection
         POINT  P0,HOLD_        ;; Nested macro must lie within the test
         LDA    P1
         OUT    A_DATA          ;; Low byte
         LDA    P1+1
         OUT    A_DATA          ;; High byte
         ELSE
         POINT  P0,HOLD_
         MVI    A, LOW P1
         OUT    A_DATA          ;; Low byte
         MVI    A, HIGH P1
         OUT    A_DATA          ;; High byte
         ENDIF
         ENDM



A>
```

The following macro code definitions implement the macro command as listed in the Macro Command Summary (Appendix D). Notice that all macros use only the A and F registers, all other working registers are unaffected. These macros are targeted for the Zilog RIO assembler.

```
S_MASK  MACRO   #1 #2 #3 #4 #5

        ; Recursive macro to get correct s-field

        COND    '#2'
        S_MASK  #2 #3 #4 #5
        ENDC
        COND    (0<#1)&(#1<6)
DLAB    DEFL    DLAB.OR.(1.SHL.(#1-1))
        ENDC
        COND    .NOT.((0<#1)&(#1<6))
        ??? #1  ; Illegal counter #
        ENDC
        ENDM

S_TYPE  MACRO   #0 #1 #2 #3 #4 #5

        ; Optimised sequence for multiple register s-field commands

        COND    (#0=20H)^(#0=40H)^(#0=60H)^(#0=0C0H)^(#0=0A0H)^(#0=80H)
DLAB    DEFL    #0
        S_MASK  #1 #2 #3 #4 #5
        LD      A,DLAB
        OUT     (A_CTRL),A
        ENDC
        COND    .NOT.((#0=20H)^(#0=40H)^(#0=60H)^(#0=0C0H)^(#0=0A0H)^(#0=80H))
        ??? #0  ; Illegal command
        ENDC
        ENDM

N_TYPE  MACRO   #1 #2

        ; Optimised sequence for single register n-type commands

        COND    (0<#1)&(#1<6)
        LD      A,#1.OR.#2
        OUT     (A_CTRL),A
        ENDC
        COND    .NOT.((0<#1)&(#1<6))
        ??? #1  ; Illegal counter
        ENDC
        ENDM

ARM     MACRO   #1 #2 #3 #4 #5  ; Arm counters, any from 1,5
        S_TYPE  20H #1 #2 #3 #4 #5
        ENDM

LOAD    MACRO   #1 #2 #3 #4 #5  ; Load counters, any from 1,5
        S_TYPE  40H #1 #2 #3 #4 #5
        ENDM

LD_ARM  MACRO   #1 #2 #3 #4 #5  ; Load 'n Arm counters, any from 1,5
        S_TYPE  60H #1 #2 #3 #4 #5
        ENDM
```

```
DISARM  MACRO   #1 #2 #3 #4 #5  ; Disarm counters, any from 1,5
        S_TYPE  0C0H #1 #2 #3 #4 #5
        ENDM

SAVE    MACRO   #1 #2 #3 #4 #5  ; Save counters, any from 1,5
        S_TYPE  0A0H #1 #2 #3 #4 #5
        ENDM

DRMSAV  MACRO   #1 #2 #3 #4 #5  ; Diserm 'n Save counters, any from 1,5
        S_TYPE  80H #1 #2 #3 #4 #5
        ENDM

SET_    MACRO   #1              ; Set single counter output #1
        N_TYPE  #1,0E8H
        ENDM

CLEAR   MACRO   #1              ; Clear single counter output #1
        N_TYPE  #1,0E0H
        ENDM

STEP    MACRO   #1              ; Step single counter output #1
        N_TYPE  #1,0F0H
        ENDM

FOUT    MACRO   #1              ; Gate FOUT on or off
        COND    '#1'='OF'
        LD      A,0EEH
        OUT     (A_CTRL),A
        ENDC
        COND    '#1'='ON'
        LD      A,0E6H
        OUT     (A_CTRL),A
        ENDC
        COND    .NOT.(('#1'='OF').OR.('#1'='ON'))
        ??? #1  ; Illegal option
        ENDC
        ENDM

RESET   MACRO                   ; Reset, load all counters
        LD      A,0FFH
        OUT     (A_CTRL),A
        LOAD    1,2,3,4,5
        ENDM

POINT   MACRO   #1 #2   ; Set data pointer register group, element

        COND    ((0<#1)&(#1<6)).OR.((#1=7)&((0.LE.#2)&(#2<4)))
        LD      A #1.OR.(#2.SHL.3)
        OUT     (A_CTRL),A
        ENDC
        COND    .NOT.(((0<#1)&(#1<6)).OR.((#1=7)&((0.LE.#2)&(#2<4))))
        ??? #1 #2
        ENDC
        ENDM
```

```
DPS     MACRO   #1      ; Set data pointer sequencing on/of
        COND    '#1'='ON'
        LD      A,0E0H
        OUT     (A_CTRL),A
        ENDC
        COND    '#1'='OF'
        LD      A,0E8H
        OUT     (A_CTRL),A
        ENDC
        COND    .NOT.(('#1'='OF').OR.('#1'='ON'))
        ??? #1  ; Illegal option
        ENDC
        ENDM

MASTER  MACRO   #1 #2 #3 #4 #5 #6 #7 #8 #9
                                        ;  Set Master register
DLAB    DEFL    #1                      ;  .day mode
DLAB    DEFL    DLAB.OR.(#2.SHL.2)      ;  .compar_1
DLAB    DEFL    DLAB.OR.(#3.SHL.3)      ;  .compar_2
DLAB    DEFL    DLAB.OR.(#4.SHL.4)      ;  .FOUT_source
DLAB    DEFL    DLAB.OR.(#5.SHL.8)      ;  .FOUT_divisor
DLAB    DEFL    DLAB.OR.(#6.SHL.12)     ;  .FOUT_gate
DLAB    DEFL    DLAB.OR.(0.SHL.13)      ;  .data_bus - 8 bit only
DLAB    DEFL    DLAB.OR.(#8.SHL.14)     ;  .data_ptr
DLAB    DEFL    DLAB.OR.(#9.SHL.15)     ;  .scaler
        POINT   CTRL_GR,MASTER_         ; Point to Master Register
        LD      A,DLAB.MOD.256
        OUT     (A_DATA),A              ; Send Low byte first
        LD      A,DLAB/256
        OUT     (A_DATA),A              ; Then High byte
        ENDM

MODE    MACRO   #0 #1 #2 #3 #4 #5 #6 #7
                                        ;  Set counter #0 mode register
        COND    (#1=3).OR.(5<#1)
        ??? #1  ; Illegal Output Control
        ENDC
        COND    .NOT.((#1=3).OR.(5<#1))
DLAB    DEFL    #1                      ;  .output
DLAB    DEFL    DLAB.OR.(#2.SHL.3)      ;  .direction
DLAB    DEFL    DLAB.OR.(#3.SHL.4)      ;  .base
DLAB    DEFL    DLAB.OR.(#4.SHL.5)      ;  .control
DLAB    DEFL    DLAB.OR.(#5.SHL.8)      ;  .source
DLAB    DEFL    DLAB.OR.(#6.SHL.12)     ;  .edge
DLAB    DEFL    DLAB.OR.(#7.SHL.13)     ;  .gate
        COND    (0<#0)&(#0<6)
        POINT   #0,MODE_                ; Point to counter #1 Mode
        LD      A,DLAB.MOD.256
        OUT     (A_DATA),A              ; Send Low byte
        LD      A,DLAB/256
        OUT     (A_DATA),A              ; Then High byte
        ENDC
        COND    .NOT.((0<#0)&(#0<6))
        ??? #0  ; Illegal counter #
        ENDC
        ENDM
```

```
LOAD_REG MACRO  #0 #1 #2        ; Set Load reg of counter #0 to #1
         POINT  #0,LOAD_
         COND   '#2'            ; Test for indirection
         LD     A,(#1)
         OUT    (A_DATA),A      ; Low byte
         LD     A,(#1+1)
         OUT    (A_DATA),A      ; High byte
         ENDC
         COND   .NOT.('#2')
         LD     A,#1.MOD.256
         OUT    (A_DATA),A      ; Low byte
         LD     A,#1/256
         OUT    (A_DATA),A      ; High byte
         ENDC
         ENDM

HOLD_REG MACRO  #0 #1 #2        ; Set Hold Reg of counter #0 to #1
         POINT  #0,HOLD_
         COND   '#2'            ; Test for indirection
         LD     A,(#1)
         OUT    (A_DATA),A      ; Low byte
         LD     A,(#1+1)
         OUT    (A_DATA),A      ; High byte
         ENDC
         COND   .NOT.('#2')
         LD     A,#1.MOD.256
         OUT    (A_DATA),A      ; Low byte
         LD     A,#1/256
         OUT    (A_DATA),A      ; High byte
         ENDC
         ENDM



A>
```

The following macro code definitions implement the macro commands as listed in the Macro Command Summary (Appendix D). Notice that all macros use only registers R0. The flag and control word register is altered but all other working registers are unaffected. These macros are targeted for the AMD MACZ assembler.

```
VAR     DLAB: OBJECT;                       % Define a dummy variable
        DLAB    ::= 0;

MACRO   S_MASK  AA;                         % Build up a dummy s_field
        BEGIN
        IF 0 LT AA AND AA LT 6
        THEN
                DLAB ::= DLAB OR (1 SHL (AA-1))
        ELSE
                CALR 0FFFFH;                % Invalid counter #
        END;

MACRO   S_TYPE  BB,BB1,BB2,BB3,BB4,BB5;

        % Optimised sequence for multiple register s-field commands

        BEGIN
        DLAB    ::= BB OR 0FF00H;
        IF NOT NULL BB5                     % Test for the parameters
        THEN
                S_MASK BB5;
        IF NOT NULL BB4
        THEN
                S_MASK BB4;
        IF NOT NULL BB3
        THEN
                S_MASK BB3;
        IF NOT NULL BB2
        THEN
                S_MASK BB2;
        IF NOT NULL BB1
        THEN
                S_MASK BB1;
        LD      R0,DLAB;
        OUT     A_CTRL,R0                   % Send the command
        END;

MACRO   N_TYPE  CC1,CC2;

        % Optimised sequence for single register n-field commands

        BEGIN
        IF (0 LT CC1) AND (CC1 LT 6)
        THEN
        BEGIN
                LD      R0,CC1 OR CC2 OR 0FF00E;
                OUT     A_CTRL,R0
        END
        ELSE
                CALR 0FFFFH;                % Illegal Counter
        END;
```

```
MACRO   ARM     DD1,DD2,DD3,DD4,DD5;    % Arm counters, any from 1,5
        BEGIN
                S_TYPE  20H,DD1,DD2,DD3,DD4,DD5
        END;

MACRO   LOAD    EE1,EE2,EE3,EE4,EE5;    % Load counters, any from 1,5
        BEGIN
                S_TYPE  40H,EE1,EE2,EE3,EE4,EE5
        END;

MACRO   LD_ARM  FF1,FF2,FF3,FF4,FF5;    % Load 'n Arm counters, any from 1,5
        BEGIN
                S_TYPE  60H,FF1,FF2,FF3,FF4,FF5
        END;

MACRO   DISARM  GG1,GG2,GG3,GG4,GG5;    % Disarm counters, any from 1,5
        BEGIN
                S_TYPE  0C0H,GG1,GG2,GG3,GG4,GG5
        END;

MACRO   SAVE    HH1,HH2,HH3,HH4,HH5;    % Save counters, any from 1,5
        BEGIN
                S_TYPE  0A0H,HH1,HH2,HH3,HH4,HH5
        END;

MACRO   DRMSAV  II1,II2,II3,II4,II5;    % Disarm 'n Save counters, any from 1,5
        BEGIN
                S_TYPE  80H,II1,II2,II3,II4,II5
        END;

MACRO   SET     JJ1;                    % Set single counter output JJ1
        BEGIN
                N_TYPE  JJ1,0E8H
        END;

MACRO   CLEAR   JJ2;                    % Clear single counter output JJ2
        BEGIN
                N_TYPE  JJ2,0E0H
        END;

MACRO   STEP    JJ3;                    % Step single counter output JJ3
        BEGIN
                N_TYPE  JJ3,0F0H
        END;

MACRO   FOUT    JJ4;                    % Gate FOUT on or off
        BEGIN
        IF JJ4 EQ OF
        THEN
        BEGIN
                LD      R0,0FFEEH;
                OUT     A_CTRL,R0
        END
        ELSE
                IF JJ4 EQ ON
                THEN
                BEGIN
                        LD      R0,0FFE6H;
                        OUT     A_CTRL,R0
                END
                ELSE
                        CALR 0FFFFH;    % Illegal request
        END;
```

```
MACRO   RESET   ;                       % Reset, load all counters
        BEGIN
        LD      R0,0FFFFH;
        OUT     A_CTRL,R0;              % Official reset
        LOAD    1,2,3,4,5;              % Clear any set TC's;
        LD      R0,1;
        OUT     A_CTRL,R0;              % Dummy set data pointer
        LD      R0,0FFEFH;
        OUT     A_CTRL,R0               % Set 16 bit data bus
        END;

MACRO   POINT   KK1,KK2 ;               % Set data pointer register group, element

        BEGIN
        IF ((0 LT KK1 AND KK1 LT 6) OR KK1 EQ 7) AND 0 LE KK2 AND KK2 LT 4
        THEN
        BEGIN
                LD      R0,KK1 OR (KK2 SHL 3) OR 0FF00H;
                OUT     A_CTRL,R0
        END
        ELSE
                CALR 0FFFFH;            % Illegal request
        END;

MACRO   DPS     P1;     % Set data pointer sequencing on/of
        BEGIN
        IF P1 EQ ON
        THEN
        BEGIN
                LD      R0,0FFE0H;
                CUT     A_CTRL,R0
        END
        ELSE
                IF P1 EQ OF
                THEN
                BEGIN
                        LD      R0,0FFE8H;
                        OUT     A_CTRL,R0
                END
                ELSE
                        CALR 0FFFFH;    % Illegal request
        END;

MACRO   MASTER  LL1,LL2,LL3,LL4,LL5,LL6,LL7,LL8,LL9;
        BEGIN
        DLAB ::= LL1;                           %  .day_mode
        DLAB ::= DLAB OR (LL2 SHL 2);           %  .compar_1
        DLAB ::= DLAB OR (LL3 SHL 3);           %  .compar_2
        DLAB ::= DLAB OR (LL4 SHL 4);           %  .FOUT_source
        DLAB ::= DLAB OR (LL5 SHL 8);           %  .FOUT_divisor
        DLAB ::= DLAB OR (LL6 SHL 12);          %  .FOUT_gate
        DLAB ::= DLAB OR (1 SHL 13);            %  .data_bus - 16 bit only
        DLAB ::= DLAB OR (LL8 SHL 14);          %  .data_ptr
        DLAB ::= DLAB OR (LL9 SHL 15);          %  .scaler
        POINT   7,2;                            % Point to Master Register
        LD      R0,DLAB;
        OUT     A_DATA,R0                       % Then High byte
        END;
```

```
MACRO   MODE_REG MM0,MM1,MM2,MM3,MM4,MM5,MM6,MM7;
        BEGIN
        IF MM1 EQ 3 OR 5 LT MM1
        THEN
                CALR 0FFFFE     % Illegal Output Control
        ELSE
        BEGIN
        DLAB ::= MM1;                       %  .output
        DLAB ::= DLAB OR (MM2 SHL 3);       %  .direction
        DLAB ::= DLAB OR (MM3 SHL 4);       %  .base
        DLAB ::= DLAB OR (MM4 SHL 5);       %  .control
        DLAB ::= DLAB OR (MM5 SHL 8);       %  .source
        DLAB ::= DLAB OR (MM6 SHL 12);      %  .edge
        DLAB ::= DLAB OR (MM7 SHL 13)       %  .gate
        END;
        IF 0 LT MM0 AND MM0 LT 6
        THEN
        BEGIN
                POINT   MM0,0;              % Point to counter MM1 Mode
                LD      R0,DLAB;
                OUT     A_DATA,R0           % Send 16 bit word
        END
        ELSE
        CALR 0FFFFH;    % Illegal counter
        END;

MACRO   LOAD_REG NN0,NN1;                   % Set Load reg of NN0 to NN1
        BEGIN
        POINT   NN0,1;                      % Notice that an explicit indirection
        LD      R0,NN1;                     %  flag (I) is not needed by Z8000
        OUT     A_DATA,R0                   % Send 16 bit word
        END;

MACRO   HOLD_REG PP0,PP1;                   % Set Hold Reg of PP0 to PP1
        BEGIN
        POINT   PP0,2;
        LD      R0,PP1;
        OUT     A_DATA,R0                   % Send 16 bit word
        END;


A>
```

The following definitions are provided for greater clarity in the C examples. These definitions are also utilized by the Am9513 evaluation program.

```
#define THEN                    /* ignore       */
#define BEGIN           {
#define END             }
#define RECORD          struct

#define CONTROL         0XDA    /* 9513 port addresses          */
#define DATA            0XD8


/*      master and counter fields       */

                        /* .FOUT_source .source values          */

#define TC_NM1          0       /* .source only                 */

#define SRC_1           1       /* Hardware source pins         */
#define SRC_2           2
#define SRC_3           3
#define SRC_4           4
#define SRC_5           5

#define GATE_1          6       /* Hardware gate pins           */
#define GATE_2          7
#define GATE_3          8
#define GATE_4          9
#define GATE_5          0XA

#define F1              0XB     /* Frequency scaler taps        */
#define F2              0XC
#define F3              0XD
#define F4              0XE
#define F5              0XF

#define BCD             1       /* .scaler modes                */
#define BINARY          0


/*      master fields           */

#define TOD_OFF         0       /* .day_mode values             */
#define TOD_50HZ        1
#define TOD_60HZ        2
#define TOD_100HZ       3

#define DISABLE         0       /* .compar1,2 values            */
#define ENABLE          1

#define ON              0       /* .FOUT_gate .data_ptr values  */
#define OFF             1

#define BUS_8           0       /* .bus_width values            */
#define BUS_16          1
```

```
/*      counter fields                  */

#define OFF_LO_TC       0       /* .output values                  */
#define ACT_HI_TC       1
#define TC_TOGGLE       2
#define OFF_OC_TC       4
#define ACT_LO_TC       5

#define UP              1       /* .direction values               */
#define DOWN            0

/*      counter mode control modes      */

#define MODE_ABC        0       /* Modes A,B,C                     */
#define MODE_DEF        1       /* Modes D,E,F                     */
#define MODE_GHI        2       /* Modes G,H,I                     */
#define MODE_JKL        3       /* Modes J,K,L                     */
#define MODE_mNO        4       /* Modes N,O - M is illegal        */
#define MODE_pQR        5       /* Modes Q,R                       */
#define MODE_Stu        6       /* Mode S                          */
#define MODE_Vwx        7       /* Mode V       */

#define RISE            0       /* .edge values                    */
#define FALL            1

#define NO_GATE         0       /* .gate values                    */
#define HL_TC_NM1       1
#define HL_NPI_GATE     2
#define HL_NM1_GATE     3
#define HL_GATE_N       4
#define LL_GATE_N       5
#define HE_GATE_N       6
#define LE_GATE_N       7


/*      load data pointer fields        */

#define MODE_           0       /* .element values                 */
#define LOAD_           1
#define HOLD_           2
#define HOLD_CYCLE      3

#define ALARM1          0       /* .element values (CTRL_GROUP)            */
#define ALARM2          1
#define MASTER          2
#define STATUS          3

#define CTRL_GROUP      7       /* .group values (inc. counters 1-5)       */

#define LOCAL           register
#define ASCICR          0XD
#define ASCILF          0XA
#define UPPER_CASE      0XDF    /* mask for upper case             */
#define RD              0       /* read or write                   */
#define WR              1
#define SEQUENCE        5       /* types of access                 */
#define NOT_SEQUENCE    6
#define LEVEL           7       /* types of gating                 */
#define EDGE            8
```

The following definitions are provided for greater clarity in the assembler examples.

```
;  Am9513 equates file for macros - Am8080/Am8085/Z80 format

        ; Master and counter Mode register functions

                                        ; FOUT source and count source definitions
TC_NM1      EQU 0       ; TC N-1 is count source only
SRC_1       EQU 1       ; Hardware source pins
SRC_2       EQU 2
SRC_3       EQU 3
SRC_4       EQU 4
SRC_5       EQU 5

GATE_1      EQU 6       ; Hardware gate pins
GATE_2      EQU 7
GATE_3      EQU 8
GATE_4      EQU 9
GATE_5      EQU 0AH

F1          EQU 0BH     ; Frequency scaler tap points
F2          EQU 0CH
F3          EQU 0DH
F4          EQU 0EH
F5          EQU 0FH

BCD         EQU 1       ; Counting modes
BINARY      EQU 0


        ; Master register functions

TOD_OFF     EQU 0       ; Time of day counting modes
TOD_50HZ    EQU 1
TOD_60HZ    EQU 2
TOD_100HZ   EQU 3

DISABLE     EQU 0       ; Comparators 1 and 2 modes
ENABLE      EQU 1

ON          EQU 0       ; FOUT gate and data pointer values
OF          EQU 1

BUS_8       EQU 0       ; 8 or 16 bit bus
BUS_16      EQU 1
```

```
        ; Counter Mode register functions

                                        ; TC output modes
OFF_LO_TC    EQU 0          ; Output inactive, low
ACT_HI_TC    EQU 1          ; Active hi terminal count pulse
TC_TOGGLE    EQU 2          ; TC toggled
OFF_OC_TC    EQU 4          ; Inactive, high impedance
ACT_LO_TC    EQU 5          ; Active lo terminal count pulse

UP           EQU 1          ; Count directions
DOWN         EQU 0

MODE_ABC     EQU 0          ; Counter mode control codes - A,B,C
MODE_DEF     EQU 1          ; Modes D,E,F
MODE_GHI     EQU 2          ; Modes G,H,I
MODE_JKL     EQU 3          ; Modes J,K,L
MODE_mNO     EQU 4          ; Modes N,L - notice Mode M is illegal
MODE_pQR     EQU 5          ; Modes Q,R
MODE_Stu     EQU 6          ; Mode S
MODE_Vwx     EQU 7          ; Mode V

RISE         EQU 0          ; Active count edge values
FALL         EQU 1

NO_GATE      EQU 0          ; Counter gating modes
EL_TC_NM1    EQU 1          ; Active Hi Level TC N-1
HL_NP1_GATE  EQU 2          ; Active Hi Level Gate N+1
HL_NM1_GATE  EQU 3          ; Active Hi Level Gate N-1
HL_GATE_N    EQU 4          ; Active Hi Level Gate N
LL_GATE_N    EQU 5          ; Active Lo Level Gate N
HE_GATE_N    EQU 6          ; Active Hi Edge Gate N
LE_GATE_N    EQU 7          ; Active Lo Edge Gate N

MODE_        EQU 0          ; Data pointer element values
LOAD_        EQU 1
HOLD_        EQU 2
HOLD_CY      EQU 3

ALARM1_      EQU 0          ; Data pointer element values (CTRL_GR)
ALARM2_      EQU 1
MASTER_      EQU 2
STATUS_      EQU 3

CTRL_GR      EQU 7          ; Data pointer group values (inc. counters 1-5)
```

```
% Am9513 equates file for macros - Z8000 format

CONST

        % Master and counter Mode register functions

                                        % FOUT source and count source definitions
TC_NM1      =   0,      % TC N-1 is count source only
SRC_1       =   1,      % Hardware source pins
SRC_2       =   2,
SRC_3       =   3,
SRC_4       =   4,
SRC_5       =   5,

GATE_1      =   6,      % Hardware gate pins
GATE_2      =   7,
GATE_3      =   8,
GATE_4      =   9,
GATE_5      =   0AH,

F1          =   0BH,    % Frequency scaler tap points
F2          =   0CH,
F3          =   0DH,
F4          =   0EH,
F5          =   0FH,

BCD         =   1,      % Counting modes
BINARY      =   0,

        % Master register functions

TOD_OFF     =   0,      % Time of day counting modes
TOD_50HZ    =   1,
TOD_60HZ    =   2,
TOD_100HZ   =   3,

DISABLE     =   0,      % Comparators 1 and 2 modes
ENABLE      =   1,

ON          =   0,      % FOUT gate and data pointer values
OF          =   1,

BUS_8       =   0,      % 8 or 16 bit bus
BUS_16      =   1,
```

```
        % Counter Mode register functions

                                        % TC output modes
OFF_LO_TC    =   0,      % Output inactive, low
ACT_HI_TC    =   1,      % Active hi terminal count pulse
TC_TOGGLE    =   2,      % TC toggled
OFF_OC_TC    =   4,      % Inactive, high impedance
ACT_LO_TC    =   5,      % Active lo terminal count pulse

UP           =   1,      % Count directions
DOWN         =   0,

MODE_ABC     =   0,      % Counter mode control codes - A,B,C
MODE_DEF     =   1,      % Modes D,E,F
MODE_GHI     =   2,      % Modes G,H,I
MODE_JKL     =   3,      % Modes J,K,L
MODE_mNC     =   4,      % Modes N,L - notice Mode M is illegal
MODE_pQR     =   5,      % Modes Q,R
MODE_Stu     =   6,      % Mode S
MODE_Vwx     =   7,      % Mode V

RISE         =   0,      % Active count edge values
FALL         =   1,

NO_GATE      =   0,      % Counter gating modes
HL_TC_NM1    =   1,      % Active Hi Level TC N-1
HL_NPI_GATE  =   2,      % Active Hi Level Gate N+1
HL_NM1_GATE  =   3,      % Active Hi Level Gate N-1
HL_GATE_N    =   4,      % Active Hi Level Gate N
LL_GATE_N    =   5,      % Active Lo Level Gate N
HE_GATE_N    =   6,      % Active Hi Edge Gate N
LE_GATE_N    =   7,      % Active Lo Edge Gate N

MODE_        =   0,      % Data pointer element values
LOAD_        =   1,
HOLD_        =   2,
HOLD_CY      =   3,

ALARM1_      =   0,      % Data pointer element values (CTRL_GR)
ALARM2_      =   1,
MASTER_      =   2,
STATUS_      =   3,

CTRL_GR      =   7;      % Data pointer group values (inc. counters 1-5)
```