# CEG-453/653 Altera Handout
# Instructions for using the ALTERA chips

This handout contains instructions for using the two Altera EPLD-5032 Erasable Programmable Logic Devices (EPLDs): Address Decoder/DTACK* Generator/VPA-VMA Generator (ADDV) and Single-Step/Watchdog Timer/Interrupt Module (SWIM). The document is organized as follows

## I. Instructions for Programming ALTERA Chips

The two custom chips (ADDV and SWIM) will be programmed into Altera EPM5032 Erasable Programmable Logic Devices (EPLDs) from Altera Definition Files (.ADF Files), which are located under `c:\ceg453\addrdec` directory on the Micron Pentium PCs. You can copy these files to your own floppy disk. The files are named DECODE.ADF and SWIM.ADF.

The procedure for preparing the chips is as follows:

1. Boot up a Micron Pentium PC in the lab that has the Altera system installed in it. The environment is Windows 95. Double click the icon for the software MaxPlusII.
2. Select Chip: From the pull down menu FILE, select PROJECT, and then select NAME. A window pops up for you to select either DECODE.ADF or SWIM.ADF. Select either one and click the OK button.
3. View Text File: From the pull down menu FILE, select HIERACHY PROJECT TOP. A window pops up for you to view (or edit) the .ADF file.
4. Assign: (1) From the pull down menu ASSIGN, select DEVICE, and use EPM5032DC-15. (2) From the pull down menu ASSIGN, select GLOBAL PROJECT LOGIC SYNTHESIS, and turn off the option of AUTOMATIC GLOBAL CLOCK.
5. Compile: From the pull down menu MAX+PLUSII, select COMPILER. Click the START button to start the compilation.
6. View Report File: Click the rpt button inside the compilation window to view the report file that summarizes the compilation results and the chip pin assignment. Or from the pull down menu FILE, select OPEN, and then select the report file ***.RPT.
7. Program Chip: Insert an EPM5032 chip into the socket on the Altera programmer by the PC. (Don't worry about the chip direction yet.) From the pull down menu MAX+PLUSII, select PROGRAMMER. A window pops up for you to program the chip. Click the BLANK-CHECK button. (You get a warning message if you placed the chip in the socket backwards.) If the chip is not blank, have the TA show you how to use the EPROM eraser (if you don't already know), and erase the chip for 15 minutes. Once the chip is blank and is put into the socket correctly, click the PROGRAM button. The chip is programmed. You may then VERIFY the chip content against the design file content (by clicking the VERIFY button.)
8. Exit from the MaxPlusII program.

## II. Using the Logic Analyzer to Test the Altera Decoder Chip

1. Make sure your breadboard is powered off.

2. Obtain three Logic Pods, wires and clips from the TA. Connect the ribbon-cable connectors to the LA4000 (metal) box and make the following connections from the POD (plastic) boxes to your breadboard. Use the suggested colors for consistency with the logic analyzer software.

| SIGNAL NAME | PROBE LEAD | | WIRE COLOR | PIN NO. |
|---|---|---|---|---|
| A0 | 0 | POD 1A | BLACK | 16 |
| A6 | 1 | | BROWN | 15 |
| A7 | 2 | | RED | 14 |
| A14 | 3 | | ORANGE | use ADDR |
| A15 | 4 | | YELLOW | 13 |
| A16 | 5 | | GREEN | 1 |
| A17 | 6 | | BLUE | use ADDR |
| A18 | 7 | | MAGENTA | use ADDR |
| A19 | 8 | POD 2A | BLACK | use ADDR |
| ADDR | 9 | | BROWN | 28 |
| RAMEN0* | 10 | | RED | 18 |
| RAMEN1* | 11 | | ORANGE | 17 |
| ROMEN0* | 12 | | YELLOW | 11 |
| ROMEN1* | 13 | | GREEN | 10 |
| ACIAEN* | 14 | | BLUE | 26 |
| PIAEN* | 15 | | MAGENTA | 19 |
| CLK | 16 | POD 3A | BLACK | 2 |
| E | 17 | | BROWN | 23 |
| AS* | 18 | | RED | 27 |
| DS* | 19 | | ORANGE | 25 |
| R/W* | 20 | | YELLOW | 5 |
| DTACK* | 21 | | GREEN | 24 |
| VPA* | 22 | | BLUE | 4 |
| --- | | | MAGENTA | |

3. Connect the GND pin of each Logic POD to your circuit's ground (i.e., with a GRAY wire and clip.)

4. Start the LA4xxx software by clicking on the "Logic Analyzer" icon on the Desktop.

5. A special setup for the Logic Analyzer has been created to facilitate testing of the Altera EPM5032 Address Decoder chip. Go to the "File" menu option, then "Load" and select the file "`c:\Program Files\LA4xxx\453DECODER.LA`."

6. Select an appropriate sampling rate from the "Clock" menu option. 100MHz (10 ns) is recommended. If you select a higher rate, you will only be able to sample 16 of the 24 channels.

7. Set up the trigger cursor by going to the "Timing" menu option and selecting the "Trig cursor to timing" option. The Trigger cursor is the thin black vertical line across the middle of the "timing view" window and will indicate the point in time when the logic analyzer triggered.

8. Select a trigger pattern by going to the "Trigger" menu option and then "Trigger word…"

- There are 16 programmable "Conditions". In the 453DECODER.LA setup, the "Condition" number indicates the only bit in the word pattern that is 0 (active-low).
    - For example, "Condition 11" on the 453DECODER.LA setup file is defined as `XXXXXXXX XXXXXXXX XXXXXXXX XXXX0XXX XXXXXXXX`, which means that, when used, it will cause a trigger when bit 11 of the PODs (RAMEN1*) is asserted. Changing different trigger conditions allows various portions of the idle loop to be examined (i.e., observe various ROM, RAM, ACIA and PIA accesses.)
        - We have created these default conditions for your convenience. You can modify them to include different combinations of 0's (active-low assertion), 1's (active-high assertion) or X's (don't care) and create your own settings (to be saved to and loaded from your working directory.)

- You can use complex sequences of "Events" to define a trigger condition. Each "Event" is associated to one of the aforementioned "Conditions".
    - For example, to trigger on an ACIA access, simply select "Condition 14" on "Event 0" and left the remaining "Events" unused.

- Use a "Count" of 1, and select the "Duration greater or equal" option to trigger when the desired condition has occurred once.
    - If you use a "Count" of N, the LA triggers on the $N^{th}$ occurrence of the condition.

9. Select a trigger mode by going to the "Trigger" menu option, then "Mode" and selecting one of the following modes:

- Single: When a trigger event occurs, the LA acquires a single buffer and stops.
- Normal: When a trigger event occurs, the LA acquires a single buffer, re-arms and repeats until stop is hit.
- Auto: Similar to Normal except that the LA will acquire regardless of the trigger event.

10. Power up your circuit.

11. Press the "GO" menu option to begin data acquisition. If the timing diagram is not displayed in a few seconds, press ``STOP'' (something is not right!).

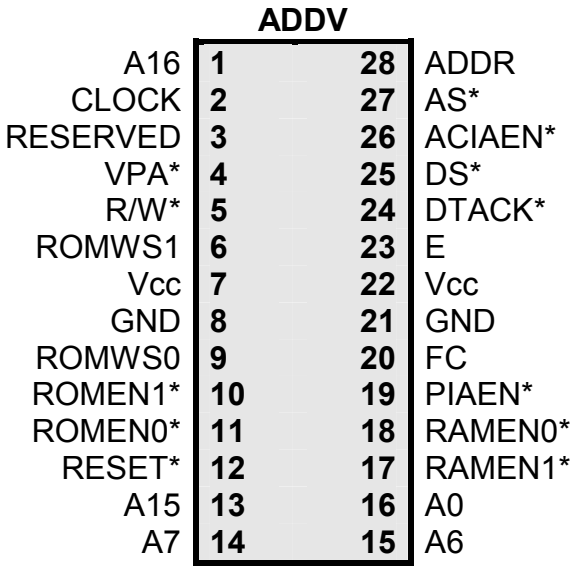
12. If you mess up, re-load the 453DECODER.LA setup; else, seek help!

13. Always turn off your circuit before disconnecting the wire clips.

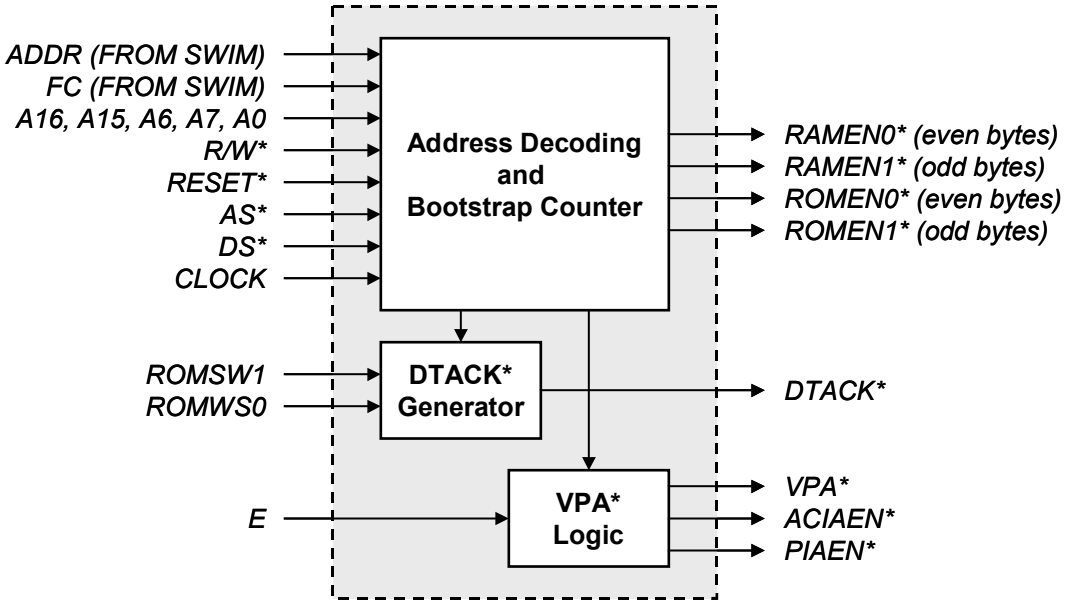
## III. Address-decoder/DTACK*-generator/VPA-VMA-generator

### III.1. Introduction:

The Address Decoder/DTACK* Generator/VPA-VMA Generator (ADDV) is an Altera EPM-5032 Erasable Programmable Logic Device (EPLD) which has been programmed to provide the device chip select signals, ROM and RAM DTACK* signal, and synchronous data transfer signals for the Motorola MC68008P8 computer of Laboratory Project 2 in CEG-453/653. The pinout of the chip is shown as follows.




A block diagram of the ADDV is shown as follows.



## III.2. Signal Definitions:

### *III.2.a. Input Signals:*

**CLOCK (Pin 2):** The 4 MHz CPU clock should be applied to this input.

**R/W* (Pin 5):** The CPU R/W* output should be applied to this input.

**ROMWS1, ROMWS0 (Pins 6, 9):** The values of these inputs determine the number of wait states inserted into ROM access bus cycles, and can also be used to inhibit DTACK* for the purpose of implementing a single-step capability. RAM accesses are always done with zero wait states, unless DTACK* is inhibited.

Table 1: ROM Wait-State And DTACK* Control

| ROMWS1 | ROMWS0 | Wait-States/DTACK* Status |
|---|---|---|
| 0 | 0 | 0 Wait States/DTACK* Enabled |
| 0 | 1 | 2 Wait States/DTACK* Enabled |
| 1 | 0 | 4 Wait States/DTACK* Enabled |
| 1 | 1 | DTACK* Inhibited |

For zero wait-state operation, both ROMWS1 and ROMWS0 should be connected to RUN* from the single-step circuit. For two wait-state operation, connect ROMWS1 to RUN* and ROMWS0 to Vcc. For four wait-states, connect ROMWS1 to Vcc and ROMWS0 to RUN.*

**RESET* (Pin 12):** The RESET* signal from the reset logic (NOT from the CPU) should be applied to this input.

**FC (Pin 20):** The FC output of the SWIM should be applied to this input. When no SWIM is available, the pin should be wired to be logic 0.

**E (Pin 23):** The CPU E clock output should be applied to this input.

**DS* (Pin 25):** The CPU DS* output should be connected to this input.

**AS* (Pin 27):** The CPU AS* output should be connected to this input.

**ADDR (Pin 28):** The ADDR output of the SWIM should be applied to this input. When no SWIM is available, the pin should be wired to be logic 1.

**A16, A15, A7, A6, A0 (Pins 1, 13, 14, 15,16):** The corresponding address outputs of the CPU should be connected to these inputs.

*III.2.b. Output Signals:*

**VPA* (Pin 4):** This output should be connected to the 68008 VPA* input. The output is asserted at the next trailing edge of the E clock after the processor starts a bus cycle with an address in the range $10000 through $13FFF or any interrupt acknowledge cycle. It returns to the inactive state at the end of the bus cycle (when AS* goes high).

**DTACK* (Pin 24):** This output should be connected to the DTACK* input of the 68008. It provides zero wait-state data transfer acknowledge for all bus cycles in the address range $0000 through $3FFF, and zero, one, or two wait-state data transfer acknowledge for read bus cycles in the address range $8000 through $BFFF. DTACK* can be inhibited during single-step operation; see the discussion of input signals ROMWS1 and ROMWS0 above.

**ROMEN1* (Pin 10):** This active-low signal should be used to enable the ROM, which contains the ODD bytes of data. It is active during read bus cycles in the address range $8001 through $BFFF and for 8 bus cycles after RESET (odd addresses only).

**ROMEN0* (Pin 11):** This active-low signal should be used to enable the ROM, which contains the EVEN bytes of data. It is active during read bus cycles in the address range $8000 through $BFFE and for 8 bus cycles after RESET (even addresses only).

**RAMEN1* (Pin 17):** This active-low signal should be used to enable the RAM, which contains the ODD bytes of data. It is active during read bus cycles in the address range $0001 through $3FFF, except for 8 bus cycles after RESET (odd addresses only).

**RAMEN0* (Pin 18):** This active-low signal should be used to enable the ROM, which contains the EVEN bytes of data. It is active during read bus cycles in the address range $0001 through $3FFF, except for 8 bus cycles after RESET (even addresses only).

**PIAEN* (Pin 19):** This active-low signal should be used to enable the PIA. It is synchronized with the E and VPA* signals by an internal VMA generator. It is active during bus cycles which address ODD locations in the range $10081 through $10087, and also certain other odd addresses in the range $10089 through $13FBE. These other addresses cause PIAEN* to be asserted because address lines A13 through A8 and A5 through A3 are ignored by this device.

**ACIAEN* (Pin 26):** This active-low signal should be used to enable the ACIA. It is synchronized with the E and VPA* signals by an internal VMA generator. It is active during bus cycles which address EVEN locations in the range $10040 through $10042, and also certain other even addresses in the range $10044 through $13F7E. These other addresses cause ACIAEN* to be asserted because address lines A13 through A8 and A5 through A2 are ignored by this device.

### *III.2.c. Reserved Pins:*

Pin marked (**RESERVED**) in Figure 1 (pin 3) is used for internal feedback on the chip. NO EXTERNAL CONNECTION should be to this pin. Doing so may interfere with the correct operation of the chip.

## III.3. Application Information:

This chip has been designed to provide device selection, DTACK* generation, and VPA* generation for a Motorola MC68008-based system, which is code-compatible with the MEX68KECB Educational Computer Board, subject to the following assumptions and limitations:

A. There are only 16K (rather than 32K) bytes of RAM, contained in two 8K X 8 static RAM devices. The TUTOR ROM code is contained in two 8K X 8 EPROM devices, with the even byte in one EPROM and the odd bytes in the other (as in ECB).

B. All RAM accesses are zero wait-state.

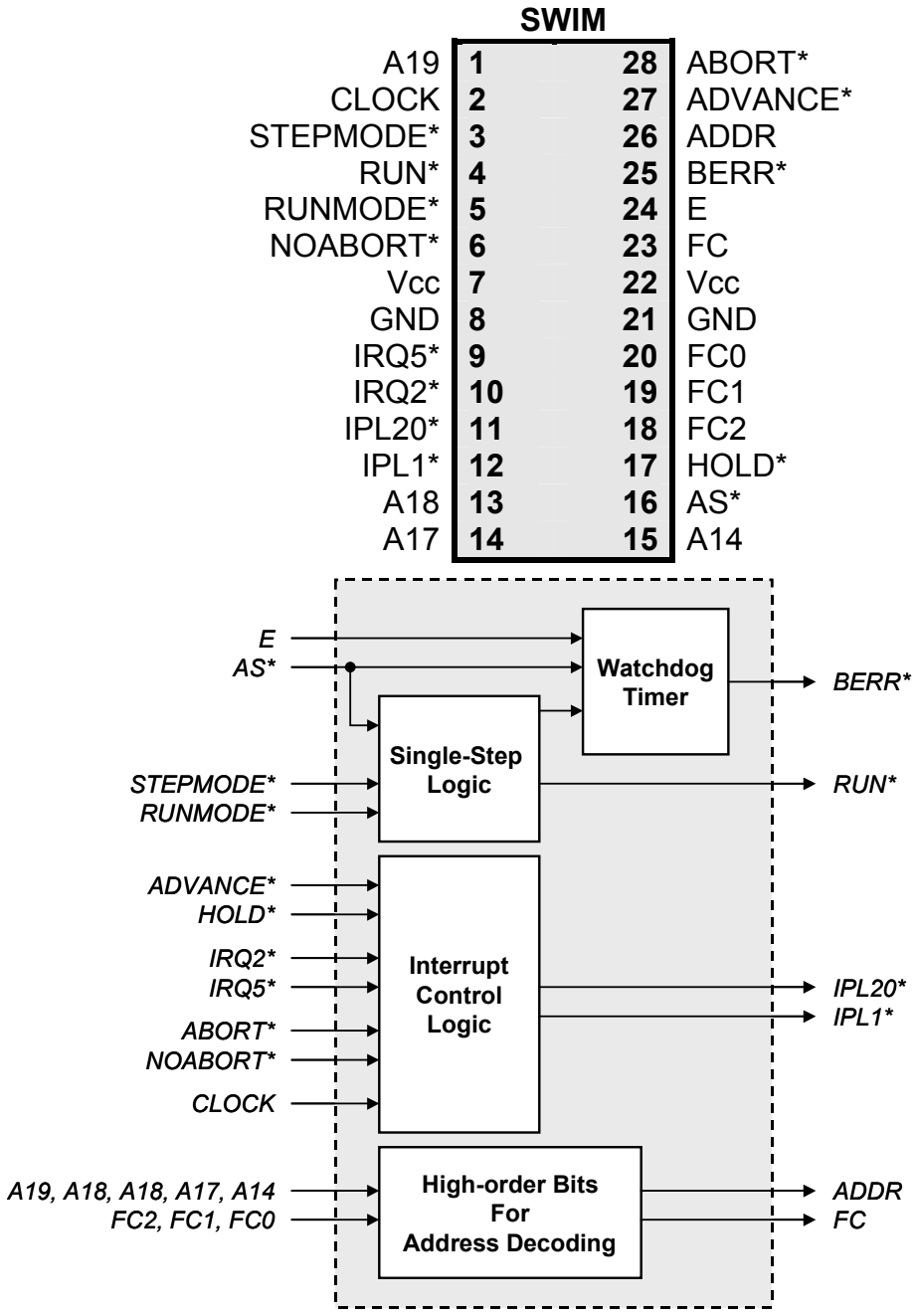
C. All interrupts are auto-vectored. This chip generates VPA* on all interrupt acknowledge cycles.

D. A Motorola MC6821 PIA (at address $10081) is used instead of the MEX68KECB's MC68230 PI/T (at address $10001). Accesses to the address range of the ECB's PI/T will be ignored by this chip, except that the VPA* output will be generated for all addresses in the range $10000 through $13FFF. The cassette interface, the printer interface, and the host port (Port 2) are not supported.

# IV. Single-Step, Watchdog-Timer and Interrupt-Module

## IV.1. Introduction:

The Single-Step/Watchdog Timer/Interrupt Module (SWIM) is an Altera EPLD-5032 Erasable Programmable Logic Device (EPLD) which has been programmed to provide the run/step control signal (RUN*), bus error signal (BERR*), and encoded interrupt priority level signals (IPL20*, IPL1*) for the Motorola MC68008P8 computer of Laboratory Project 2 in CEG-453/653. It includes the switch debouncers for the RUN/STEP switch, the single-step switch, and the software abort switch. The pinout of the chip is shown as follows.



## IV.2. Signal Definition:

### *IV.2.a. Input Signals:*

**E (pin 24):** The CPU E clock output should be applied to this input.

**STEPMODE*, RUNMODE* (Pins 3, 5):** Theses inputs should be pulled up to Vcc through appropriate resistors (1K to 4.7K ohms). A single-pole, double-throw switch should be provided which will connect RUNMODE* to ground in one position (the RUN position) and connect STEPMODE* to ground in the other position (the STEP position). When this RUN/STEP switch is in the RUN position, the RUN* output will always be low. When it is in the STEP position, the RUN* output will be controlled by the single-step switch described below. See the description of the RUN* output below for further information.

**HOLD*, ADVANCE* (Pins 17, 27):** These pins should be pulled up to Vcc through appropriate resistors (1K to 4.7K ohms). A spring-loaded SPDT switch should be provided which in the normal position connects HOLD* to ground, and in the depressed position connects ADVANCE* to ground. When the RUN/STEP switch is in the STEP position, this single-step switch will control the RUN* output. See the description of the RUN* output below for further information.

**AS* (Pin 16):** The CPU AS* output should be connected to this input.

**IRQ5*, IRQ2* (Pins 9, 10):** The active-low interrupt request signals from devices associated with level 5 and level 2 interrupts respectively should be connected to these inputs. Since the outputs of these devices will probably be open-collector outputs, they should be pulled up to Vcc through appropriate resistors (1K to 4.7K ohms).

**NOABORT*, ABORT* (Pins 6, 28):** These inputs should be pulled up to Vcc through appropriate resistors (1K to 4.7K ohms). A spring-loaded SPDT switch should be provided which in the normal position connects NOABORT* to ground, and in the depressed position connects ABORT* to ground. This software abort switch, when pressed, acts as a level 7 interrupt input to the interrupt control logic. See the description of the IPL20* and IPL1* outputs below for further information.

**CLOCK (Pin 2):** The 4 MHz CPU clock should be applied to this input.

**A19, A18, A17, A14 (Pins 1, 13, 14, 15):** The corresponding address outputs of the CPU should be connected to these inputs.

**FC2, FC1, FC0 (Pins 18, 19, 20):** The FC2, FC1, FC0 outputs of the CPU should be applied to these inputs.

*IV.2.b. Output Signals:*

**RUN* (Pin 4):** This output is always low when the RUN/STEP switch is in the RUN position. When the RUN/STEP switch is in the STEP position, it will go high at the beginning of each new bus cycle, and go low when the single-step switch is pressed. It can be used to inhibit the return of DTACK* to the CPU by memory devices. The use of RUN* as an input to the 45301-5.00 Address Decoder/DTACK* Generator/VPA* Generator Module is described in the handout which describes that module.

**BERR* (Pin 25):** This output is always high when the RUN/STEP switch is in the STEP position. When the RUN/STEP switch is in the RUN position, it will go low if the AS* input remains low for four rising edges of the E input, and return to high when AS* goes high. It should be connected to the BERR* input of the MC68008; if connected in this way, it will cause the processor to trap to the bus error routine whenever the address strobe remains active for more that four rising edge of the E clock (31 to 40 system clock cycles).

**IPL20*, IPL1* (Pins 11, 12):** These outputs represents the (active-low) encoded level of the highest priority requesting interrupt, and should be connected to the IPL2/0* and IPL1* inputs of the MC68008 respectively. Table 1 shows the values which these outputs will assume for each combination of the inputs IPL5* and IPL2* and the position of the software abort switch.

Table 1: Interrupt Control Logic Truth Table

| IRQ2* | IRQ5* | ABORT* switch | IPL20* | IPL1* |
|---|---|---|---|---|
| X | X | PRESSED | 0 | 0 |
| X | 0 | NOT PRESSED | 0 | 1 |
| 0 | 1 | NOT PRESSED | 1 | 0 |
| 1 | 1 | NOT PRESSED | 1 | 1 |

The IPL20* and IPL1* outputs shown in the table become effective on the second rising edge of the CLOCK input following the establishment of the input conditions shown.

**ADDR (Pin 26):** This output should be connected to the ADDR input of the Address Decoder/DTACK* Generator/VPA* Generator Module.

**FC (Pin 23):** This output should be connected to the FC input of the Address Decoder/DTACK* Generator/VPA* Generator Module.

## IV.3. Application Information:

This chip has been designed to provide single-step logic, BERR* generation, and interrupt control logic for a Motorola MC68008-based system which is code-compatible with the MEX68KECB Educational Computer Board. It s intended to be used as a companion chip to the ADV. If these two chips are used in the CEG-453/653 Laboratory 2 computer, only the microprocessor, memory and I/O chips, reset logic, and RS-232 drivers and receivers are required to complete the project. No additional TTL logic, except that used in the reset module, should be required.

## V. ADDV Altera Definition File

```
CEG 453
EPM5032
DECODER/WAIT-STATE GENERATOR/VPA-VMA GENERATOR

OPTIONS:
        % The TURBO option causes the chip to consume %
        % more power, but operate faster.  Delay from %
        % input to output is reduced by about 30 nsec %
        % by setting this option.                     %

        TURBO = ON

PART:
        EPM5032

INPUTS:
        % The external names of all input pins are    %
        % listed in the INPUTS section.               %

        A16, A15, A7, A6, A0, RESET*, AS*, DS*, RW*,
        FC, ROMWS1, ROMWS0, CLOCK, E, ADDR


OUTPUTS:
        % The external names of all output pins are   %
        % listed in the OUTPUTS section.              %

        RAMEN0*, RAMEN1*, ROMEN0*, ROMEN1*, ACIAEN*,
        PIAEN*, VPA*, DTACK*

NETWORK:
        % Relate internal input names to external in- %
        % put names.  Names which contain * must be   %
        % changed for internal representation.        %

        ADDR = INP (ADDR)
        A16 = INP (A16)
        A15 = INP (A15)
        A7 = INP (A7)
        A6 = INP (A6)
        A0 = INP (A0)
        ASB = INP (AS*)
        DSB = INP (DS*)
        RWB = INP (RW*)
        RESETB = INP (RESET*)
        FC = INP (FC)
        ROMWS1 = INP (ROMWS1)
        ROMWS0 = INP (ROMWS0)
        E = INP (E)
        CLOCK = INP(CLOCK)

        % Signals to be used as asynchronous clocks   %
        % must be buffered. CLKB is an asynchronous   %
        % clock buffer.  En is defined later in the   %
```

```
        % Boolean Equations section.                  %

        ASBa = CLKB (ASB)
        Ea = CLKB (En)

        % The next several subsections define the     %
        % output and/or feedback names associated     %
        % with various logic Macrocells.  The out-    %
        % put configurations used are as follows:     %

        %   CONF: Combinational output, no feedback.  %
        %     OUTPUT = CONF (INPUT,ENABLE)            %
        %   COIF: Combinational output, I/O feedback. %
        %     OUTPUT,FEEDBACK = COIF (INPUT,ENABLE)   %
        %   NORF: No output, registered feedback.     %
        %     FEEDBACK = NORF (D, CLK, CLEAR, PRESET) %
        %   NOJF: No output, JK feedback.             %
        %     FBACK = NOJF (J, CLK, K, CLEAR, PRESET) %

        % Default enable signal is Vcc (always ena-   %
        % bled.  For this chip, presets are not im-   %
        % plemented, and must be grounded.  Note that %
        % output names can include the character *,   %
        % but feedback names cannot.  Input names are %
        % defined later in the Boolean Equations sec- %
        % tion.                                       %

        %     DEVICE SELECTS     %

        RAMEN0* = CONF (RAMEN0c,)
        RAMEN1* = CONF (RAMEN1c,)
        ROMEN0*,ROMEN0f = COIF (ROMEN0c,)
        ROMEN1*,ROMEN1f = COIF (ROMEN1c,)
        ACIAEN* = CONF (ACIAENb,)
        PIAEN* = CONF (PIAENb,)

        %     BOOT CIRCUIT     %

        QA = NORF (DA, ASBa, RESET, GND)
        QB = NORF (DB, ASBa, RESET, GND)
        QC = NORF (DC, ASBa, RESET, GND)
        BOOTB = NORF (BOOTBd, ASBa, RESET, GND)

        %     DTACK* GENERATOR     %

        WS0 = NORF (VCC, CLOCK, ROMSELc, GND)
        WS1 = NORF (WS1d, CLOCK, ROMSELc, GND)
        WS2 = NORF (WS2d, CLOCK, ROMSELc, GND)
        DTACK* = CONF (DTACKc,)

        %     VPA*/VMA GENERATOR     %

        VPA = NOJF (VPAj,Ea,GND,ASB,GND)
        VMA = NORF (VMAd,CLOCK,ASB,GND)
        VPA* = CONF (VPAn,)

EQUATIONS:
```

```
        % This section provides the Boolean Equations %
        % which relate the variables to each other.   %
        % the symbol / before a variable or term      %
        % means "the complement of."                  %

        %     DEVICE SELECTS     %

        RAMEN0c = /(BOOTB * /DSB * /FC * ADDR
                * /A16 * /A15 * /A0);
        RAMEN1c = /(BOOTB * /DSB * /FC * ADDR
                * /A16 * /A15 * A0);
        ROMEN0c = /((/BOOTB * /DSB * /A0) + (/ASB * /FC
                * RWB * ADDR * /A16 * A15 * /A0));
        ROMEN1c = /((/BOOTB * /DSB * A0) + (/ASB * /FC
                * RWB * ADDR * /A16 * A15 * A0));
        ACIAENc = /(/DSB * /FC * ADDR
                * A16 * /A15 * /A7 * A6 * /A0);
        PIAENc = /(/DSB * /FC * ADDR
               * A16 * /A15 * A7 * /A6 * A0);

        %     BOOT CIRCUIT     %

        DA = /QA;
        DB = (QA * /QB) + (/QA * QB);
        DC = (QA * QB) + QC;
        BOOTBd = (QA * QB * QC) + BOOTB;
        RESET = /RESETB;

        %     DTACK* GENERATOR     %

        DTACKRAMc = /(BOOTB * (/DSB + /ASB * /RWB)
                  * ADDR * /A16 * /A15);
        ROMSELc = ROMEN0f * ROMEN1f;    WS1d = WS0;
        WS2d = WS1;
        DTACKc = (DTACKRAMc * (/WS0 + ROMWS1 + ROMWS0)
               * (/WS1 + ROMWS1 + /ROMWS0)
               * (/WS2 + /ROMWS1 + ROMWS0))
               + (ROMWS0 * ROMWS1);

        %     VPA*/VMA GENERATOR     %

        VPAj = ((ADDR * A16 * /A15) + FC) * /ASB;
        VPAn = /VPA;
        VMAd = (/ACIAENc + /PIAENc) * VPA;
        ACIAENb = ACIAENc + /VMA;
        PIAENb = PIAENc + /VMA;
        En = /E;

END$
```

## VI. SWIM Altera Definition File

```
CEG 453
EPM5032
SINGLE-STEP/WATCHDOG TIMER/INTERRUPT MODULE

OPTIONS:
        % Setting the TURBO option causes the chip to %
        % consume more power, but operate faster.     %

        TURBO = ON

PART:
        EPM5032

INPUTS:
        % External names of input pins are listed.    %

        E, CLOCK, NOABORT*, AS*, ADVANCE*, RUNMODE*,
        STEPMODE*, ABORT*, HOLD*, IRQ2*, IRQ5*, A19,
        A18, A17, A14, FC0, FC1, FC2

OUTPUTS:
        % External names of output pins are listed.   %

        IPL1*, IPL20*, RUN*, BERR*, ADDR, FC

NETWORK:

        % Internal input names are defined.  External %
        % names containing * must be changed.         %


        A19 = INP(A19)
        A18 = INP(A18)
        A17 = INP(A17)
        A14 = INP(A14)
        FC0 = INP(FC0)
        FC1 = INP(FC1)
        FC2 = INP(FC2)
        RUNMODEb = INP (RUNMODE*)
        STEPMODEb = INP (STEPMODE*)
        ADVANCEb = INP (ADVANCE*)
        HOLDb = INP (HOLD*)
        ABORTb = INP (ABORT*)
        NOABORTb = INP (NOABORT*)
        ASb = INP (AS*)
        IRQ2b = INP (IRQ2*)
        IRQ5b = INP (IRQ5*)
        CLOCK = INP(CLOCK)
        E = INP(E)

        % Define asynchronous clock buffers for Ad-  %
        % dress Strobe and an internal signal (used  %
        % in the single-step circuit)                %
```

```
        ASc = CLKB (AS)
        STEPc = CLKB (QSTEPf)

        % Define macrocell outputs. See DECODEV5.ADF %
        % for configurations and other information.  %
        % Other configurations used here are:        %

        %   NOCF: No output, combinational feedback. %
        %   RONF: Registered output, no feedback.    %

        %     SINGLE-STEP CIRCUIT     %

        QSTEPSf = NOCF (QSTEPS)
        QNSTEPMf = NOCF (QNSTEPM)
        QSTEPf = NOCF (QSTEP)
        QRUN = NORF (VCC, STEPc, ASb, GND)
        QSTEPMODE = NORF (QSTEPM, ASc, QNSTEPMf, GND)
        RUN* = CONF (RUNb,)

        %     WATCHDOG TIMER MODULE     %

        QT1 = NORF (AS, E, WCLf, GND)
        QT2 = NORF (QT1, E, WCLf, GND)
        QT3 = NORF (QT2, E, WCLf, GND)
        BERR = NORF (QT3, E, WCLf, GND)
        BERR* = CONF (BERRb,)
        WCLf = NOCF (WCLR)

        %     INTERRUPT ENCODER MODULE     %

        QSWAf = NOCF (QSWA)
        Q2b = NORF (IRQ2b, CLOCK, GND, GND)
        Q5b = NORF (IRQ5b, CLOCK, GND, GND)
        QABT = NORF (QSWA, CLOCK, GND, GND)
        IPL20* = RONF (IPL20b, CLOCK, GND, GND,)
        IPL1* = RONF (IPL1b, CLOCK, GND, GND,)

        % ADDRESS DECODING MODULE for decodev5.adf    %

        ADDR = CONF(ADDRint,)
        FC = CONF(FCint,)


EQUATIONS:
        % Relate signals to each other through    %
        % Boolean Equations.                      %

        %    ADDRESS DECODING MODULE     %

        ADDRint = /A19 * /A18 * /A17 * /A14;
        FCint = FC0 * FC1 * FC2;

        %     SINGLE STEP MODULE     %

        AS = /ASb;
        QSTEPS = /(ADVANCEb *  QNSTEPS);
        QNSTEPS = /(QSTEPSf* HOLDb);
```

```
        QSTEP = QSTEPS * QSTEPM;
        QSTEPM = /(STEPMODEb * QNSTEPMf);
        QNSTEPM = /(QSTEPM * RUNMODEb);
        RUNb = /(/QSTEPMODE + QRUN);

        %     WATCHDOG TIMER MODULE     %

        BERRb = /BERR;
        WCLR = ASb + /QNSTEPMf;

        %     INTERRUPT ENCODER MODULE     %

        QSWA = /(ABORTb * QNSWA);
        QNSWA = /(QSWAf * NOABORTb);
        IPL20b = Q5b * /QABT;
        IPL1b= /QABT * /(/Q2b * Q5b);

END$
```

## VII. MC68008P8, ADDV and SWIM pin-out reference

**ADDV**

| | | | |
|---|---|---|---|
| A16 | 1 | 28 | ADDR |
| CLOCK | 2 | 27 | AS* |
| RESERVED | 3 | 26 | ACIAEN* |
| VPA* | 4 | 25 | DS* |
| R/W* | 5 | 24 | DTACK* |
| ROMWS1 | 6 | 23 | E |
| Vcc | 7 | 22 | Vcc |
| GND | 8 | 21 | GND |
| ROMWS0 | 9 | 20 | FC |
| ROMEN1* | 10 | 19 | PIAEN* |
| ROMEN0* | 11 | 18 | RAMEN0* |
| RESET* | 12 | 17 | RAMEN1* |
| A15 | 13 | 16 | A0 |
| A7 | 14 | 15 | A6 |

**SWIM**

| | | | |
|---|---|---|---|
| A19 | 1 | 28 | ABORT* |
| CLOCK | 2 | 27 | ADVANCE* |
| STEPMODE* | 3 | 26 | ADDR |
| RUN* | 4 | 25 | BERR* |
| RUNMODE* | 5 | 24 | E |
| NOABORT* | 6 | 23 | FC |
| Vcc | 7 | 22 | Vcc |
| GND | 8 | 21 | GND |
| IRQ5* | 9 | 20 | FC0 |
| IRQ2* | 10 | 19 | FC1 |
| IPL20* | 11 | 18 | FC2 |
| IPL1* | 12 | 17 | HOLD* |
| A18 | 13 | 16 | AS* |
| A17 | 14 | 15 | A14 |

**MC68008P8**

| | | | |
|---|---|---|---|
| *A3* | 1 | 48 | *A2* |
| *A4* | 2 | 47 | *A1* |
| *A5* | 3 | 46 | *A0* |
| *A6* | 4 | 45 | *FC0* |
| *A7* | 5 | 44 | *FC1* |
| *A8* | 6 | 43 | *FC2* |
| *A9* | 7 | 42 | *IPL2/0** |
| *A10* | 8 | 41 | *IPL1** |
| *A11* | 9 | 40 | *BERR** |
| *A12* | 10 | 39 | *VPA** |
| *A13* | 11 | 38 | *E* |
| *A14* | 12 | 37 | *RESET** |
| *VCC* | 13 | 36 | *HALT** |
| *A15* | 14 | 35 | *GND* |
| *GND* | 15 | 34 | *CLK* |
| *A16* | 16 | 33 | *BR** |
| *A17* | 17 | 32 | *BG** |
| *A18* | 18 | 31 | *DTACK** |
| *A19* | 19 | 30 | *R/W** |
| *D7* | 20 | 29 | *DS* |
| *D6* | 21 | 28 | *AS** |
| *D5* | 22 | 27 | *D0* |
| *D4* | 23 | 26 | *D1* |
| *D3* | 24 | 25 | *D2* |