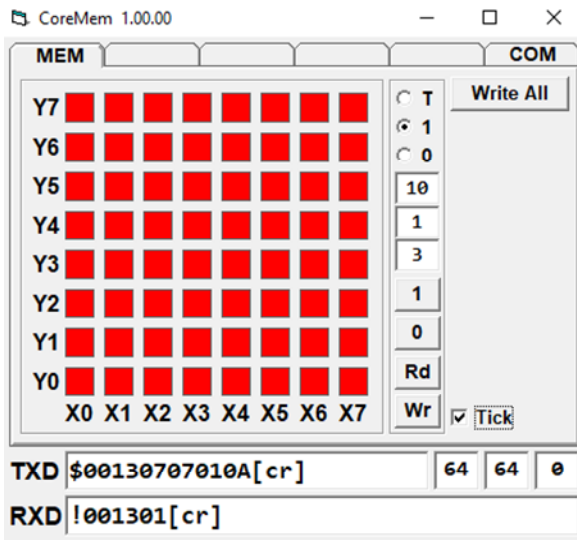


64 BIT MAGNETIC MEMORY SYSTEM



64 bit core matrix
1747C x 1

$C_{FILT} = 100pF$

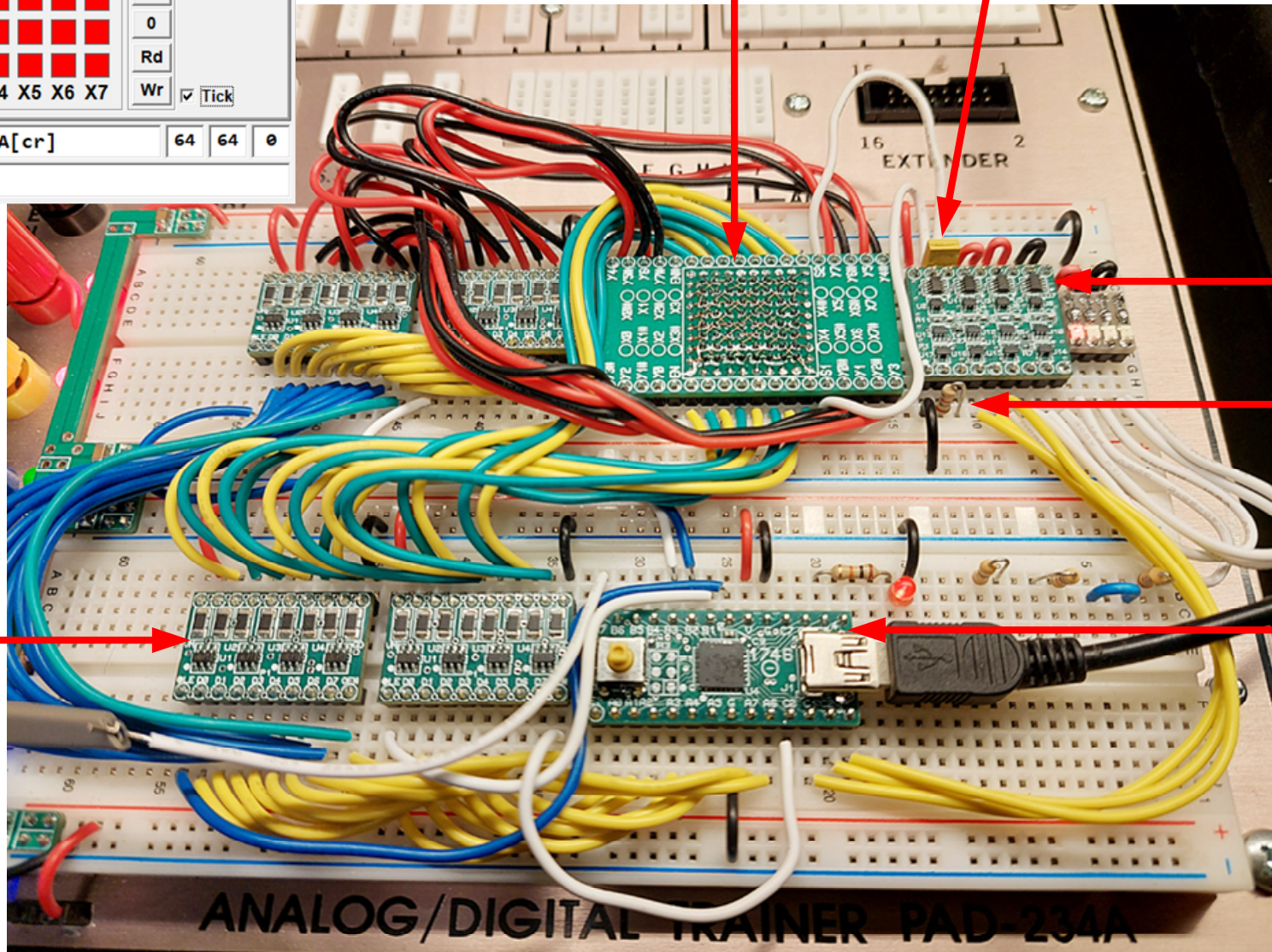
Y Drivers
1749 x 2
 $R = 10\Omega$

X Drivers
1749 x 2
 $R = 10\Omega$

Sense Amp
1752 x 1

$RSET = 1.2K$
 $= 0.95V/1.2K$
 $= 792\mu A$
 $= 7.92mV$

Controller
1746P x 1



TEST UI

The screenshot shows the CoreMem 1.00.00 application window. It features a 'MEM' tab with an 8x8 grid of red squares representing memory cells, labeled Y7-Y0 on the left and X0-X7 on the bottom. A 'COM' tab is on the right, containing a 'Write All' button, radio buttons for 'T' (Toggle), '1' (Set), and '0' (Clear), and input fields for '10' (Hold time), '1' (Repeat count), '3' (Cycle delay), and '1' (Set all boxes). Below these are buttons for 'Rd' (Read) and 'Wr' (Write), and a 'Tick' checkbox. At the bottom, 'TXD' and 'RXD' fields show hexadecimal data, and three status boxes display '64', '64', and '0'. Red arrows point from text labels to various UI elements: 'Clicking each bit cell writes to the matrix using the selected command' points to a cell in the grid; 'Command each bit in sequence' points to the 'Write All' button; 'Command: Toggle, Set or Clear' points to the radio buttons; 'Hold time, uS' points to the '10' field; 'Command repeat count' points to the '1' field; 'Write all cycle delay, n x 100mS' points to the '3' field; 'Set all boxes to Red = "1"' points to the '1' field; 'Clear all boxes to Green = "0"' points to the '0' field; 'Read entire matrix' points to the 'Rd' button; 'Write entire matrix' points to the 'Wr' button; 'Audible tick for Write All function' points to the 'Tick' checkbox; 'Sense amp reported "No change"' points to the first '64' box; 'Sense amp reported "Change"' points to the second '64' box; and 'Command count' points to the '0' box.

\$0013xxyyssh[cr]

\$0013 Matrix write command
 xx X coordinate, 00-07
 yy Y coordinate, 00-07
 ss Core state, 00 or 01
 hh Hold time, uS

!0013ss[cr]

!0013 Matrix write response
 ss Core state, 00 or 01

TEST FW (IN CCS PICC)

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////  
void xy_drv_init(void){  
  
    driver_fwd[0] = 0b00000001;  
    driver_fwd[1] = 0b0000100;  
    driver_fwd[2] = 0b00010000;  
    driver_fwd[3] = 0b01000000;  
    driver_rev[0] = 0b00000010;  
    driver_rev[1] = 0b00001000;  
    driver_rev[2] = 0b00100000;  
    driver_rev[3] = 0b10000000;  
  
}  
  
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////  
//Clears all drivers to 0x00 = Float  
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////  
void driver_clr(void){  
  
    //All driver latches written at once  
    dataout = 0x00;  
    blat = 0x0F;  
    delay_cycles(1);  
    blat = 0x00;  
  
    //When high, all drivers see latch as tri-state, i.e. pulled down.  
    //Now that the latches are initialized, it's safe to let them take control.  
    driveren_n = 0;  
  
}
```

```

////////////////////////////////////
//Write to individual cores in the matrix
//00000xxx,00000yyy,000000m,dddddddd
//m      1      Sets X=1/X#=0
//      0      Sets X=0/X#=1
//
//After delay, driver returns to X=0/X#=0 and Y=0/Y#=0
//All unselected driver pairs set to 00, i.e. float mode
////////////////////////////////////
unsigned int1 xy_drv(unsigned int8 x,
                    unsigned int8 y,
                    unsigned int8 mode,
                    unsigned int8 delay){
unsigned int1 datain;

//Scope trigger high
trigger = 1;

//Reset the sense amplifier latches
senserst_n = 0;
delay_cycles(1);
senserst_n = 1;

//Forward is X = 1 / X# = 0
//Reverse is X = 0 / X# = 1
if (mode)
    dataout = driver_fwd[x & 3];
else
    dataout = driver_rev[x & 3];

//Drivers are 4 channels per byte so decide which driver to address
if (x & 4)
{
    x74 = 1;
    delay_cycles(1);
    x74 = 0;
}
else
{
    x30 = 1;
    delay_cycles(1);
    x30 = 0;
}

//Forward is Y = 1 / Y# = 0
//Reverse is Y = 0 / Y# = 1
if (mode)
    dataout = driver_fwd[y & 3];
else
    dataout = driver_rev[y & 3];

//Drivers are 4 channels per byte so decide which driver to address
if (y & 4)
{
    y74 = 1;
    delay_cycles(1);
    y74 = 0;
}
else
{
    y30 = 1;
    delay_cycles(1);
    y30 = 0;
}

//Hold the X and Y current to allow the core to flip.
//This usually happens about 5uS after the current is turned on
if (delay)
    delay_us(delay);

//Read the sense amplifier state (Since demo only uses 1 pin, not on data bus)
datain = sense0;

//All off, write all drivers to 0x00 at the same time
dataout = 0x00;
blat = 0x0F;
delay_cycles(1);
blat = 0x00;

//Scope trigger low
trigger = 0;

return datain;
}
////////////////////////////////////

```