The Family 2 Programmer

By

Rick Holmes

Electronics Engineer

Electronic Product Testing Center (EPTC)

Defense Supply Center Columbus (DSCC)

Defense Logistics Agency (DLA)

[richard.m.holmes@dla.mil](mailto:richard.m.holmes@dla.mil)

(614) 692-9870

Introduction

One of the missions of the Electronics Product Testing Center (EPTC) is the PROM/ROM program. This is a program where DLA buys and stocks blank Programmable Read Only Memory(PROMS) devices and upon customer demand programs them with unique software to make Read Only Memory (ROM) devices. This converts the devices from the PROM’s NSN to the ROM’ NSN. ROMs made from a common PROM are collected into “Families.” The common PROM is called a “head of Family”. There are currently 283 different heads of Families. Consequently, 283 different PROMs are stocked by NSN. Each Family may have from just a few members to hundreds of members. There are over 15,000 members collectively. Converting blank PROMs into unique ROMs eliminates the need to stock 15,000 individual NSNs.

Up until October, 2011, the PROM/ROM mission was performed at the Defense Depot Hill Utah (DDHU), Hill AFB. At that time, all the PROM/ROM equipment was transferred to EPTC including a Data I/O 29B programmer which is used to program Families 54, 55, 56, 57 and 58, a Spectrum Dynamics 500 programmer which is used to program Family 2 and a Data I/O Unisite Xpi programmer which is used to program all the other remaining Families. Some other programmers such as a Data I/O System 19, Data I/O 222 and ChipMaster 6000 were also transferred, but they are not used. However, according to the cognizant engineers at DDHU, the Spectrum Dynamics programmer has not been operational since 2003 and it is uneconomical to repair even if someone could be found to repair it. This left EPTC without the capability to program Family 2 devices.

Devices are sent to Lansdale for programming

The head of Family is a Harris HM-0512-8 NSN 5962-00-003-2304. According to the HM-0512-8 specification sheet, this device is programmable by a Data I/O System 19 with a 909-1054-3 card set. EPTC has the System 19 but not this particular card set. Quotes for the card set were solicited from vendors but all the responses were in excess of $6,500 and all the card sets were used with no guarantee of working.

Next, the possibility of designing an adapter for an existing programmer was investigated. However, because the HM-0512-8 requires a negative programming voltage while other similar devices use a positive programming voltage, designing, building and using an adapter would be unwieldy and may induce damage to an otherwise serviceable programmer.

Next, the possibility of designing and building a programmer from scratch was investigated. From examination of the HM-0512-8 data sheet which included instructions for programming the devices, it was decided that the datasheet contained sufficient detail that designing and building a programmer was the most viable option.

Programming Algorithm

The relays are configured to connect the device as shown in Figure X. To address a particular word in memory, the appropriate switches are set to the equivalent binary word where a logical low is -5 VDC and a logical 1 is an open circuit. A programming pulse is applied to the fusing circuit which creates a -7 VDC pulse to the first bit to be changed from a logical low to a logical high. Skipping any bit which is to remain low, repeat the programming pulse until the 8 bit byte has been completely programmed. Only one bit is programmed at a time. Increment the address and repeat the process to program the next 8 bit byte

Figure X. Programming Circuit

Figure X. Family 2 Programmer Setup

Program Listing

/\*

Harris (Lansdale) 0512 Programmer.

Data is read in from a text file via a comm port from a PC and into a 64 element string array. The data file consists of 8 ASCII characters of "0" or "1" representing a byte. The bytes are in consecutive order from 0 to 63. To address a particular word in the memory, the input relays are controlled by the address counter to close the relay contacts to apply -5.0 VDC to make a logic 0 or leave the relay contacts open to make a logic 1. Each bit of the byte is tested to see if it is a 1. If it is, a -5 VDC programming voltage is applied for 700 msec. Otherwise, it is skipped and the next bit is tested until all 8 bits of the byte are tested. Once the current byte has been programmed, the address counter is incremented and the process is repeated for the next byte.

\*/

char\* menuStrings[] = {"\n\n\n Harris 0512 Programmer (Family 2)\n\n",

" 1) Load RAM from file",

" 2) Load RAM from master device",

" 3) Blank Check",

" 4) Program device",

" 5) Checksum RAM",

" 6) Checksum device",

" 7) Read device contents.",

" 8) Read the contents of RAM",

" Enter 1 - 8 ->"};

char incomingByte;

String myString[] = {"00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000",

"00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000",

"00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000",

"00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000",

"00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000",

"00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000",

"00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000",

"00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000", "00000000"};

void loadRAMfromFile(void);

void loadRAMfromDevice(void);

void blankCheck(void);

void programDevice(void);

void checksumRAM(void);

void verifyDevice(void);

void readRAM(void);

void setup() {

// initialize the digital pins as outputs.

pinMode(2, OUTPUT); // D0 programming only, not for reading from prom

pinMode(3, OUTPUT); // D1

pinMode(4, OUTPUT); // D2

pinMode(5, OUTPUT); // D3

pinMode(6, OUTPUT); // D4

pinMode(7, OUTPUT); // D5

pinMode(8, OUTPUT); // D6

pinMode(9, OUTPUT); // D7

pinMode(10, OUTPUT); // programming pulse

pinMode(11, OUTPUT); // clear the address generator

pinMode(12, OUTPUT); // increment the address generator

/\* A0 thru A5 and E1 and E2 should always be tri stated when applying programming voltage.

E1 and E2 should always remain high when reading the prom \*/

pinMode(22, OUTPUT); // A0 address for reading prom only

pinMode(23, OUTPUT); // A1

pinMode(24, OUTPUT); // A2

pinMode(25, OUTPUT); // A3

pinMode(26, OUTPUT); // A4

pinMode(27, OUTPUT); // A5

pinMode(28, OUTPUT); // E1

pinMode(29, OUTPUT); // E2

pinMode(30, OUTPUT); // address buffer enable.

/\* B0 thru B7 should always be tristated when applying programming pulses \*/

pinMode(31, INPUT); // B0

pinMode(32, INPUT); // B1

pinMode(33, INPUT); // B2

pinMode(34, INPUT); // B3

pinMode(35, INPUT); // B4

pinMode(36, INPUT); // B5

pinMode(37, INPUT); // B6

pinMode(38, INPUT); // B7

pinMode(39, OUTPUT); // data buffer enable

pinMode(40, OUTPUT); // G2' open when programming, closed to ground when reading.

/\* relay 41 and 42 should NEVER be turned on at the same time. If they are, the -5 VDC will

be shorted to ground

\*/

pinMode(44, OUTPUT);

digitalWrite(11,HIGH); // clear

digitalWrite(12,LOW); // clock

digitalWrite(30, HIGH); // tristate the address buffer

digitalWrite(39, HIGH); // tristate the data buffer

digitalWrite(40, LOW); // G2' open

digitalWrite(44, LOW); // G1 ground

Serial.begin(2400); // set serial port to 2400 baud

}

void loop() {

digitalWrite(11,LOW); // clear the address generator (synchronous counter)

delay (10);

digitalWrite(12, HIGH); // since this is a synchronous counter, give it a clock pulse

delay (10);

digitalWrite(12,LOW);

digitalWrite(11, HIGH);

for (int i=0; i < 63; i++){ // because the outputs of the counter are now inverted, the clock must be

digitalWrite(12, HIGH); // run up to 63 (11 1111) so that the address relays are turned off

delay (10);

digitalWrite(12,LOW);

delay (10);

}

for (int m=0; m<10; m++){ // output the menu

Serial.println(menuStrings[m]);}

while(Serial.available() ==0){}; // wait fo a character to come in

incomingByte = Serial.read(); // read the character

switch (incomingByte) { // test the character to see which menu item was selected and

case '1': // go to appropriate subroutine

loadRAMfromFile();

break;

case '2':

loadRAMfromDevice();

break;

case '3':

blankCheck();

break;

case '4':

programDevice();

break;

case '5':

checksumRAM();

break;

case '6':

checksumDevice();

break;

case '7':

readDevice();

break;

case '8':

readRAM();

break;

default:

Serial.println("\n Invalid input. Enter 1 -8 ->");} // invalid input was entered

}

void loadRAMfromFile(){

char temp;

boolean ok = false;

Serial.println("Loading RAM from PC file. Start file transfer from PC");

Serial.flush();

for (int i=0; i < 64; i++){ // go through each address

for(int j =0; j < 8; j++){

while(ok == false){

while (Serial.available() == 0){}

temp = Serial.read();

if ((temp == '0') || (temp == '1')){

ok = true;}

}

ok = false;

myString[i].setCharAt(j, temp);

}

Serial.println(myString[i]);}

Serial.flush();

Serial.println("\n Done");

return;}

void loadRAMfromDevice(){

int readIn = 0;

Serial.println("Loading RAM from master.");

digitalWrite(28, HIGH); // Set the enable E1

digitalWrite(29, HIGH); // Set the enable E2

digitalWrite(44, LOW); // Make sure -5 VDC is off on G1

digitalWrite(40, HIGH); // Connect G2' to ground

for (int m = 0; m < 6; m++){

digitalWrite(m+22, LOW);}

for (int j = 2; j < 10; j++){

digitalWrite(j, LOW);} // make sure B0-B7 of PROM is connected to Arduino data 31 thru 38

digitalWrite(30, LOW); // Enable address and data buffers

digitalWrite(39, LOW);

for (int a5 = 0; a5 < 2; a5++){ // These loopa generate the PROM address

digitalWrite(27, a5);

for (int a4 = 0; a4 < 2; a4++){

digitalWrite(26, a4);

for (int a3 = 0; a3 < 2; a3++){

digitalWrite(25, a3);

for (int a2 = 0; a2 < 2; a2++){

digitalWrite(24, a2);

for (int a1 = 0; a1 < 2; a1++){

digitalWrite(23, a1);

for (int a0 = 0; a0 < 2; a0++){

digitalWrite(22, a0);

delay(10);

for (int k = 0; k < 8; k++){ //This loop reads the PROM data bit bybit from B0 to B7

readIn = digitalRead(38-k);

if(readIn == HIGH){

myString[((32\*a5)+(16\*a4)+(8\*a3)+(4\*a2)+(2\*a1)+(a0))].setCharAt(k, '1');}

else{

myString[((32\*a5)+(16\*a4)+(8\*a3)+(4\*a2)+(2\*a1)+(a0))].setCharAt(k, '0');}}

Serial.println(myString[((32\*a5)+(16\*a4)+(8\*a3)+(4\*a2)+(2\*a1)+(a0))]);

}}}}}}

digitalWrite(30, HIGH); // tristate the address buffer

digitalWrite(39, HIGH); // tristate the data buffer

Serial.println("Done.");

return;}

void blankCheck(){

boolean blank = true;

int readIn;

Serial.println("Checking if device is blank.");

digitalWrite(28, HIGH); // Set the enable E1

digitalWrite(29, HIGH); // Set the enable E2

digitalWrite(44, LOW); // Make sure -5 VDC is off on G1

digitalWrite(40, HIGH); // Connect G2' to ground

for (int m = 0; m < 6; m++){

digitalWrite(m+22, LOW);}

for (int j = 2; j < 10; j++){

digitalWrite(j, LOW);} // make sure B0-B7 of PROM is connected to Arduino data 31 thru 38

digitalWrite(30, LOW); // enable the address buffer

digitalWrite(39, LOW); // enable the data buffer

for (int a5 = 0; a5 < 2; a5++){

digitalWrite(27, a5);

for (int a4 = 0; a4 < 2; a4++){

digitalWrite(26, a4);

for (int a3 = 0; a3 < 2; a3++){

digitalWrite(25, a3);

for (int a2 = 0; a2 < 2; a2++){

digitalWrite(24, a2);

for (int a1 = 0; a1 < 2; a1++){

digitalWrite(23, a1);

for (int a0 = 0; a0 < 2; a0++){

digitalWrite(22, a0);

delay(5);

for (int k = 0; k < 8; k++){

readIn = digitalRead(38-k);

if(readIn == HIGH){

blank = false;}}

}}}}}}

digitalWrite(30, HIGH); // tristate the address buffer

digitalWrite(39, HIGH); // tristate the data buffer

if (blank){

Serial.println("Done. Device is blank.");}

else{

Serial.println("Done. WARNING! Device not blank!");}

return;}

void programDevice(){

Serial.println("Programming device.");

digitalWrite(11, LOW);

delay(5);

digitalWrite(12, HIGH);

delay(5);

digitalWrite(12, LOW);

delay(5);

digitalWrite(11, HIGH);

digitalWrite(30, HIGH); // tristate the address buffer

digitalWrite(39, HIGH); // tristate the data buffer

digitalWrite(40, LOW); // disconnect G2' from ground and leave open

digitalWrite(44, HIGH); // Connect G1 to -5 VDC

for (int i=0; i < 64; i++){ // for testing only

// for (int i=0; i < 64; i++){ // go through each address

for (int j=7; j > -1; j--){ // go through each character(bit) in the byte

if (myString[i].charAt(j)== '1'){ // test to see if the bit needs to be programmed

digitalWrite((7-j+2), HIGH); // add 2 because byte starts on digital output 2

delay(10);

digitalWrite(10, LOW); // turn on the programming pulse

// delay(25);

delay(700); // wait the appropriate amount of time

digitalWrite(10, HIGH); // turn off programming pulse

delay(10);

digitalWrite((7-j+2), LOW);

delay(10);}

}

if (i != 63){ // don't increment the last time

digitalWrite(12, HIGH);

delay (10);

digitalWrite(12,LOW);

delay (50);}

}

digitalWrite(44, LOW); // Turn off the - 5 VDC

Serial.println("Done.");

return;}

void checksumRAM(void){

int checkSum = 0;

int byteVal;

int index;

Serial.println("Performing checksum on RAM");

for (int i=0; i < 64; i++){ // go through each address

byteVal = 0;

index = 128;

for (int j=0; j < 8; j++){ // go through each character(bit) in the byte

if (myString[i].charAt(j)== '1')

byteVal = byteVal + index;

index = index/2;}

checkSum = checkSum + byteVal;}

Serial.println("Done. Checksum is ");

Serial.print(checkSum, HEX);

Serial.println(" ");

return;}

void checksumDevice(void){

int checkSum = 0;

int byteVal;

int index;

int readIn;

Serial.println("Performing checksum on Device");

digitalWrite(28, HIGH); // Set the enable E1

digitalWrite(29, HIGH); // Set the enable E2

digitalWrite(44, LOW); // Make sure -5 VDC is off on G1

digitalWrite(40, HIGH); // Connect G2 prime to ground

for (int m = 0; m < 6; m++){

digitalWrite(m+22, LOW);}

for (int j = 2; j < 10; j++){

digitalWrite(j, LOW);} // make sure B0-B7 of PROM is connected to Arduino data 31 thru 38

digitalWrite(30, LOW); // enable the address buffer

digitalWrite(39, LOW); // enable the data buffer

for (int a5 = 0; a5 < 2; a5++){

digitalWrite(27, a5);

for (int a4 = 0; a4 < 2; a4++){

digitalWrite(26, a4);

for (int a3 = 0; a3 < 2; a3++){

digitalWrite(25, a3);

for (int a2 = 0; a2 < 2; a2++){

digitalWrite(24, a2);

for (int a1 = 0; a1 < 2; a1++){

digitalWrite(23, a1);

for (int a0 = 0; a0 < 2; a0++){

digitalWrite(22, a0);

delay(5);

byteVal = 0;

index = 128;

for (int k = 0; k < 8; k++){

readIn = digitalRead(38-k);

if(readIn == HIGH){

byteVal = byteVal + index;}

index = index/2;}

checkSum = checkSum + byteVal;

}}}}}}

digitalWrite(30, HIGH); // tristate the address buffer

digitalWrite(39, HIGH); // tristate the data buffer

Serial.println("Done. Checksum is ");

Serial.print(checkSum, HEX);

Serial.println(" ");

return;}

void readDevice(void){

int readIn;

String tempString[] = {"00000000","11111111"};

Serial.println("Reading device contents.");

digitalWrite(28, HIGH); // Set the enable E1

digitalWrite(29, HIGH); // Set the enable E2

digitalWrite(44, LOW); // Make sure -5 VDC is off on G1

digitalWrite(40, HIGH); // Connect G2 prime to ground

for (int m = 0; m < 6; m++){

digitalWrite(m+22, LOW);}

for (int j = 2; j < 10; j++){

digitalWrite(j, LOW);} // make sure B0-B7 of PROM is connected to Arduino data 31 thru 38

digitalWrite(30, LOW);

digitalWrite(39, LOW);

for (int a5 = 0; a5 < 2; a5++){

digitalWrite(27, a5);

for (int a4 = 0; a4 < 2; a4++){

digitalWrite(26, a4);

for (int a3 = 0; a3 < 2; a3++){

digitalWrite(25, a3);

for (int a2 = 0; a2 < 2; a2++){

digitalWrite(24, a2);

for (int a1 = 0; a1 < 2; a1++){

digitalWrite(23, a1);

for (int a0 = 0; a0 < 2; a0++){

digitalWrite(22, a0);

delay(5);

for (int k = 0; k < 8; k++){

readIn = digitalRead(38-k);

if(readIn == HIGH){

tempString[1].setCharAt(k, '1');}

else{

tempString[1].setCharAt(k, '0');}}

Serial.println(tempString[1]);

}}}}}}

digitalWrite(30, HIGH);

digitalWrite(39, HIGH);

Serial.println("Done.");

return;}

void readRAM(void){

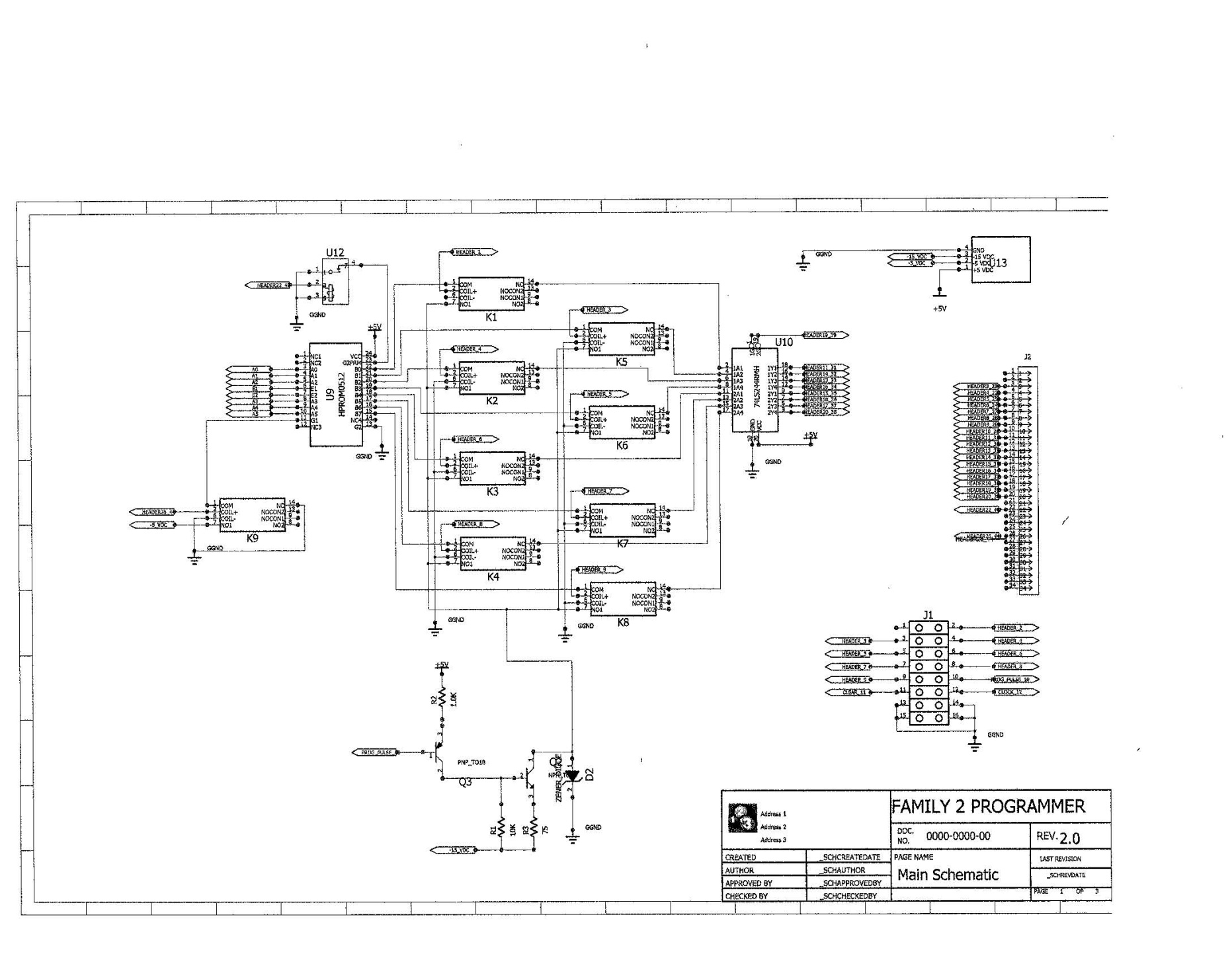
Serial.println("Sending contents of RAM to serial port.");

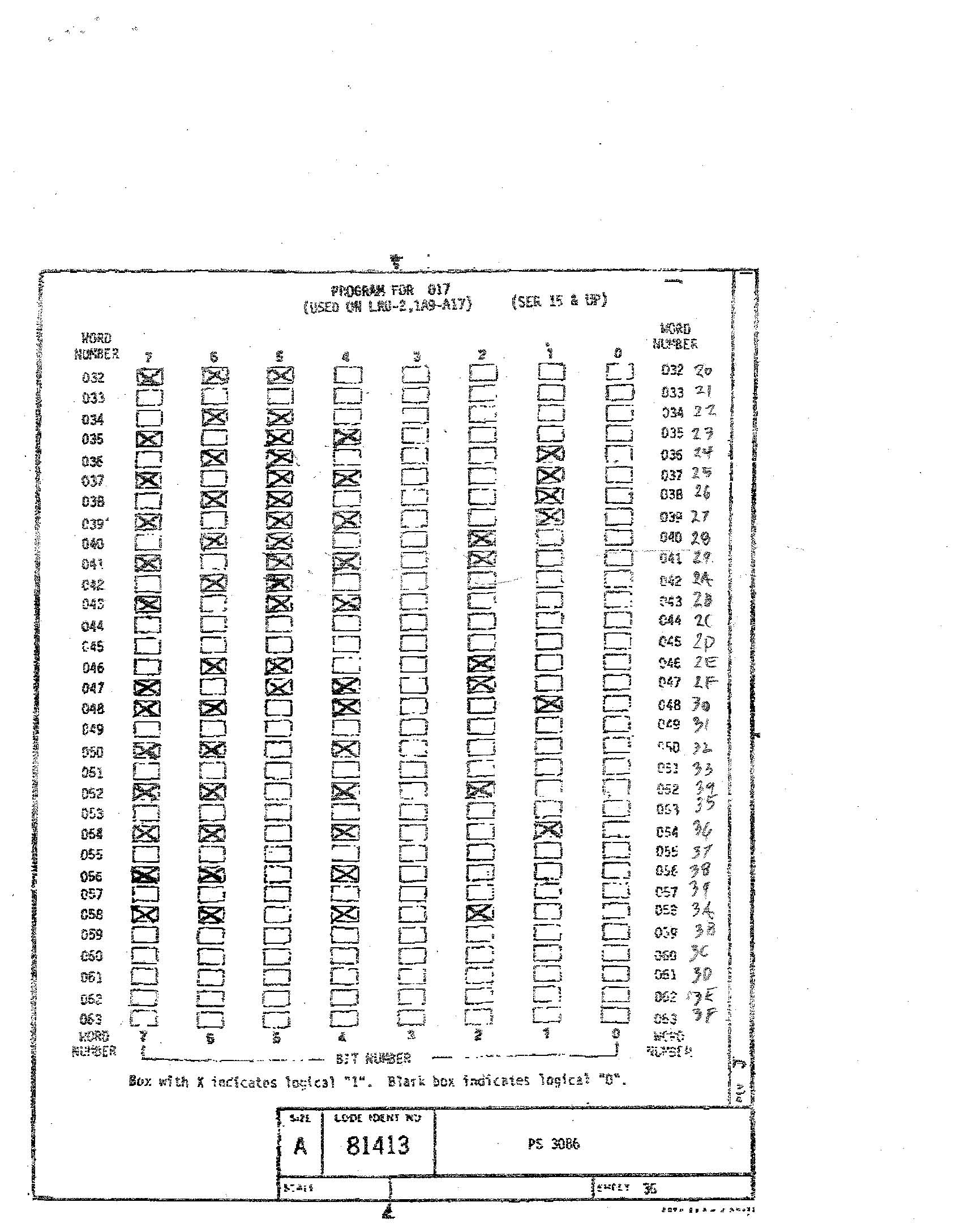
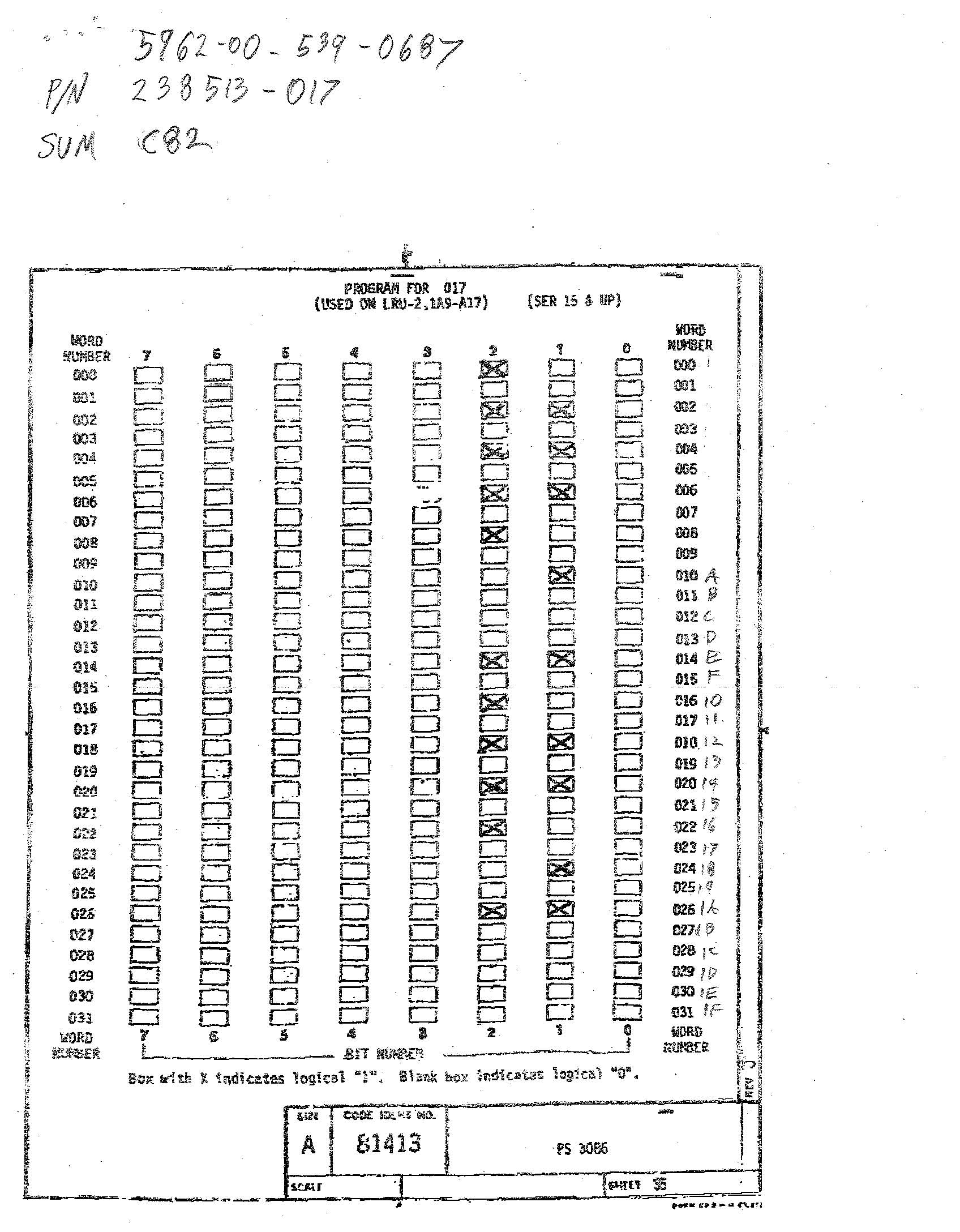
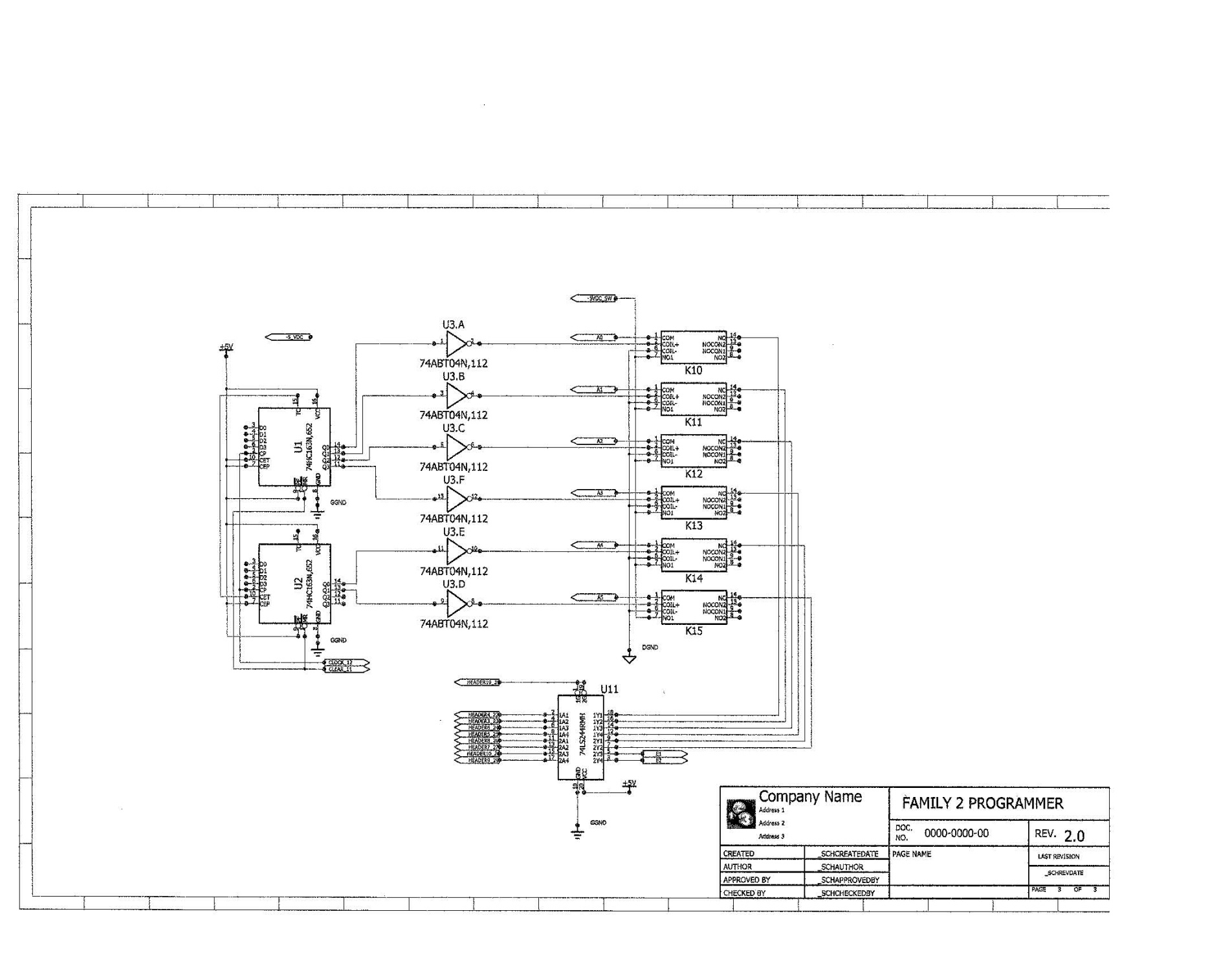
for (int i=0; i < 64; i++){ // go through each address

Serial.println(myString[i]);}

Serial.println("Done.");

return;}





Data File for NSN 5962-00-539-0687 P/N 238513-017 developed from Drawing 81413 Sheet 35 and 36

00000100

00000000

00000110

00000000

00000110

00000000

00000110

00000000

00000100

00000000

00000010

00000000

00000000

00000000

00000110

00000000

00000100

00000000

00000110

00000000

00000110

00000000

00000100

00000000

00000010

00000000

00000110

00000000

00000000

00000000

00000000

00000000

11100000

00000000

01100000

10110000

01100010

10110010

01100010

10110010

01100100

10110100

01100000

10110000

00000000

00000000

01100100

10110100

11010010

00000000

11010000

Truth Table for the Hex to 7 Segment Decoder

Hex to 7 Segment Decoder Listing

Title Decoder

Pattern decoder.pds

Revision A

Author R. Holmes

Company EPTC

Date 03/26/2012

CHIP decoder PAL16L8

;PINS 1 2 3 4 5 6 7 8 9 10

B0 B1 B2 B3 NC NC NC NC NC GND

;PINS 11 12 13 14 15 16 17 18 19 20

NC a bb c d e f g NC NC

EQUATIONS

a = /((/B3\*/B2\*/B1\*B0)M + (/B3\*B2\*/B1\*/B0) + (B3\*/B2\*B1\*B0) + (B3\*B2\*/B1\*B0)

bb = /((/B3\*B2\*/B1\*B0) + (/B3\*B2\*B1\*/B0) + (B3\*/B2\*B1\*B0) + (B3\*B2\*/B1\*/B0) + (B3\*B2\*B1)

c = /((/B3\*/B2\*/B1\*/B0) + (B3\*B2\*/B1\*/B0) + (B3\*B2\*B1)

d = /((/B3\*/B2\*/B1\*B0) + (/B3\*B2\*/B1\*/B0) + (/B3\*B2\*/B1\*B0) + (/B3\*B2\*B1\*B0) + (B3\*/B2\*/B1\*B0) + (B3\*/B2\*B1\*/B0) + (B3\*B2\*B1\*B0)

e = /((/B3\*/B2\*/B1\*B0) + (/B3\*/B2\*B1\*B0) + (/B3\*B2\*/B1) + (/B3\*B2\*B1\*B0) + (B3\*/B2\*/B1\*B0))

f = /((/B3\*/B2\*/B1\*B0) + (/B3\*/B2\*B1) + (/B3\*/B2\*B1) + (/B3\*B2\*B1\*B0) + (B3\*B2\*/B1\*B0))

g = /((/B3\*/B2\*/B1) + (/B3\*B2\*B1\*B0) + (B3\*B2\*/B1\*/B0)

