

# In-System Programming of Phase Change Memory

## Case Study for Programming Micron's M25PX64 Phase Change Memory with Teradyne's TestStation In-Circuit Tester

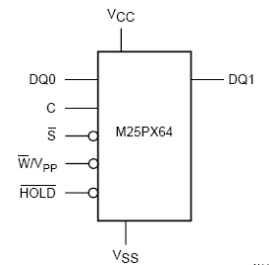
### Application Overview

Phase Change Memory (PCM) is an innovative memory technology that has fast write speeds and "no-erase" capability, attributes making it functionally equivalent to non-volatile DRAM. Because PCM is a thermally activated technology, and cannot tolerate reflow temperatures, they cannot be pre-programmed prior to PCB Assembly – therefore, they are ideal candidates for programming during the in-circuit test process. This application brief describes activities that were performed by Teradyne field support to program a Micron M25PX64 Phase Change Memory on Teradyne's TestStation in-circuit test system.

#### Overview of Micron M25PX64

The M25PX64 is an 8 pin device that functions as a 64-Mbit (8 Mbits x 8) serial flash memory. It has advanced write protection mechanisms, accessed by a high speed SPI-compatible bus and supports two high-performance dual input/output instructions:

The memory can be programmed 1 to 256 bytes at a time, using the page program instruction. The memory is organized as 128 sectors that are further divided into 16 subsectors each (2048 subsectors in total). The memory can be erased one 4-Kbyte subsector at a time, one 64-Kbyte sector at a time, or as a whole. It can be write protected by software using a mix of volatile and non-volatile protection features, depending on the application needs.



For complete details on the electrical, mechanical, and programming characteristics refer to the **Micron M25PX64** data sheet.

## Device Test & Programming Strategy

Unlike lower performance in-circuit testers that need to add third party device programming hardware and software to perform efficient programming of ISP devices on their systems, the advanced capabilities of Teradyne's standard TestStation hardware and software are well suited to effectively and efficiently test and program the latest PCM and ISP technologies. The benefits of using the standard instrumentation instead of a third party solution are lower costs, faster development, simpler fixturing, reduced training, and better test portability.

Based on an analysis of the M25PX64 device, it was decided to use a 5 step strategy when creating the Teradyne TestStation in-circuit device test.

1. **Check Manufacturing ID** - The first step checks that the correct Manufacturing ID code can be read from the device. This step verifies that the right device has been placed on the PCB and that the tester can communicate with it. If this test fails, a diagnostic message is reported to the operator and the follow-on erase, programming, and verify steps are skipped.
2. **Perform Sector Erase** – The second step uses the sector erase command to erase (set all bits to '1') the locations that are to be programmed. The data sheet for the M25PX64 device specifies a typical sector erase time of 0.7 seconds and a maximum sector erase time of 3 seconds. To account for the time differences between devices, and to keep the erase time at the absolute minimum, the program loops and waits only until the device erase status bit indicates that the sector erase cycle is complete.
3. **Sector Blank Check** – The third step verifies that the memory locations that are to be programmed are properly erased. It does this by performing a fast read of all locations to make sure that all 4 Megabits are erased. If the test fails, the operator is notified and the programming and verify step is skipped.
4. **Program Memory Data** – The fourth step is to program the 4Mbit locations with the target application data. Teradyne's Deep Serial Memory (DSM) instrument is used to store the application data. The benefit of using DSM is that the application data is kept separate from the programming algorithms and it can be changed at any time without changing the test program or incurring any production down time. In addition, the large capacity of the DSM instrument allows the software to load the programming data once and keep it in memory so there are no delays related to "re-loading" the pin memory every time the test is run.
5. **Verify Programming Data** – The final step performs a read of the memory locations to verify that they have been properly programmed. All 4 Mbits of data is read and compared against the application programming data that is stored in the DSM instrument.

The test is structured so that steps 2, 3, and 4 can be skipped when the device has already been programmed and the operator only wants to verify that the correct device is on the board and it has the correct application data.

## Tester Timing Setup

The TestStation has a sophisticated Digital Timing Subsystem that was utilized to optimize both the programming code and the test execution time of the M25PX64 device. Prior to running the test, a dedicated Clock pin and multiple timing sets were defined to support the specific test timing requirements:

- **Dedicated Clock Pin** – A dedicated tester clock pin is assigned to the **Serial Clock** pin of the device. Clock pins, part of Teradyne's Clock/Sync/Trigger (CST) instrument, can operate independently of D/S timing at speeds up to 20MHz.

- **Multiple Timing Sets** – To accommodate the different timing of the write, read, and erase cycles of the device; multiple timing sets are defined prior to execution of the test. All three timing sets were programmed to last 200ns. In the first timing set the Clock pin is programmed to stay high for the 200ns test step and the drivers are programmed drive at the start of the test step and the sensors are programmed to sense at 150ns. In the second timing set, the Clock pin is programmed to go low at 0ns and high at 100ns, the drivers are programmed to drive at 50ns, and the sensors are programmed to sense at 150ns. Finally, in the third timing set, the Clock pin is programmed to go high at 0ns and low at 25ns while the drivers are programmed to drive at 50ns and the sensors are programmed to sense 150ns

When creating the test, the programmer can easily specify which of the three Timing Sets to use for each test vector by placing a TS(1), TS(2), or TS(3) identifier on the test vector.

## Data Compression Features

The TestStation high speed controller and software supports a number of data compression features that really come in handy when developing tests for programmable logic devices that may require large amounts of data. The data compression features of the Teradyne TestStation that were utilized while creating the test for the M25PX64 device include the following.

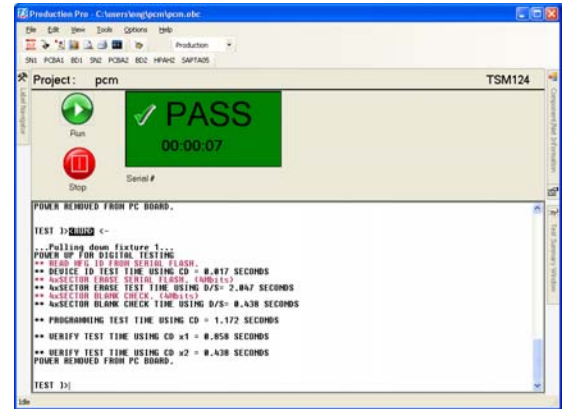
- **Loops / Nested Loops** – Loops allow programmers to create efficient programming of repetitive steps. Nested loops can be used to create complex timing sequences using very few test steps. Nested loops are used in this test to loop through the memory addresses during verify and write cycles and to wait for status bits indicating completion of erase and write cycles.
- **Tables and Hold (HD) pin states** – Tables can be used to store vector data that can be referenced from the main test. It is useful when combined with Loops to apply changing vector data in a minimum amount of test steps. During the Programming Burst for the M25PX64, it was possible to represent 8,192 unique memory addresses with only 288 Table test vectors. The HD pin condition is used to force the D/S pins to hold whatever test vector condition they were assigned to in the Table.
- **Subroutines** – Subroutines are useful when test code must be repeated several times during a test procedure. The common code can be written once, stored in a Subroutine, and then referenced as needed by the test procedure. Subroutines simplify the amount of code that programmers need to write and it makes the test procedure easier to support. For this test, a common Data Polling subroutine is called after each Sector Erase command to determine when the erase sequence is finished.
- **Deep Serial Memory** – The Deep Serial Memory is an option on the TestStation Integrated Controller Board that offers up 1024 Megabytes of memory that can be dynamically assigned to any pin in the system. It is very useful for boundary scan and memory test applications that may require large amounts of data. Essentially, the DSM stores vector state data which can be used along with the Loop feature to execute very long test sequences in very few test programming steps. In this application, the DSM is used to store 4Mb of data that the programmer would like to program into the M25PX64 device.

For more information on any of these features, refer to the *TestStation Test Language Reference* or *TestStation Test Library Programming* manuals.

## Test Times

The screenshot below identifies the time that was measured to execute the five step test strategy for the M25PX64 device. Here is the summary of the test times:

1. Read Manufacturer's ID	0.017 seconds
2. Erase 4Mb of Data	2.047 seconds
3. Perform 4Mb Blank Check	0.438 seconds
4. Program 4 Mb Data	1.172 seconds
5. Verify 4Mb of Data	0.438 seconds
<hr/>	
Total Execution Time	4.112 seconds



A comparison of test times with an alternative ICT solution that utilized a separate hardware Utility Card with a dedicated Serial Peripheral Interface showed that the TestStation native solution ran significantly faster. The total execution time for the Utility SPI solution was 29.74 seconds compared to the total execution time of 4.11 seconds shown above for the TestStation (over 7 times faster!). These faster test times can be realized on the Teradyne tester because of the advanced hardware and software capabilities of the TestStation digital subsystem.

## Hardware / Software Requirements

The application code developed for the M25PX64 device will run on any Teradyne TestStation or legacy 228X test system that is running software version 5.8.0 or greater. The TestStation UltraPin test systems are recommended for the increased accuracy of the D/S pins and the SafeTest Protection features, like real-time backdrive current monitoring that can detect potential isolation problems that could cause device damage or reduce program reliability.

## Application Example

Below is the example code that was used to test and verify the M25PX64 on a Teradyne test system.

```
/* Define Clock and D/S Timing Sets */
SET CLOCK AT(1)
    NINC=8,8,8
    TINC=25N
    DST(D0,S6),(D2,S6),(ED2,S6)
    SIGNAL(CDA=H0),(CDA=L0,H4),(CDA=H0,L1);

T3=TIME(); /* Record Start Time in variable T3 */
```

```

/*****
/** The next routine is a short-hand coding method to set up two **/
/** bit string variables that that will be imported into the **/
/** programming Burst. The variables will contain a counting **/
/** pattern that will be utilized to access all the memory **/
/** locations that will be programmed. **/
*****/
INIT: LET LOAD_ADDR_BS_LO = (0);
      LET LOAD_ADDR_BS_HI = (0);

      LET BITPOS_LO = 1;
      LET BITPOS_HI = 1;

FORLOOP1_T1: FOR LOAD_ADDR = 0 TO 255 STEP 1 DO
[ /* Turn Floating Point Number into a Bit String */
  ADDR_BS_LO = (0);
  ADDR_BS_LO = FLTOBS(LOAD_ADDR,1,8);

  FORLOOP2_T1: FOR N=8 TO 1 STEP -1 DO
    [ /* Set the appropriate bits in the 2048 bit Bitstring */
      IF ADDR_BS_LO(N) THEN BITSET(LOAD_ADDR_BS_LO,BITPOS_LO);
      LET BITPOS_LO = BITPOS_LO + 1;
    ];
];

FORLOOP1_T2: FOR LOAD_ADDR = 0 TO 31 STEP 1 DO
[ /* Turn Floating Point Number into a Bit String */
  ADDR_BS_HI = (0);
  ADDR_BS_HI = FLTOBS(LOAD_ADDR,1,5);

  FORLOOP2_T2: FOR N=5 TO 1 STEP -1 DO
    [ /* Set the appropriate bits in the 32 bit Bitstring */
      IF ADDR_BS_HI(N) THEN BITSET(LOAD_ADDR_BS_HI,BITPOS_HI);
      LET BITPOS_HI = BITPOS_HI + 1;
    ];
];
];

```

```

/* Step 1 - Read Manufacturer's ID */

U1_ID1:
WRITE '%033%M** READ MFG ID FROM SERIAL FLASH. %033%:N';

    BURST NAME='U1_ID';
START_MAIN:
    FAST;
IC(CSN,SI) IH(CSN) IL(SI);;

/* RDID COMMAND 1001 1111*/
IL(CSN);
TS(2) IH(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IH(SI);
TS(2) IH(SI); TS(2) IH(SI); TS(2) IH(SI); TS(2) IH(SI);

/** PCM ID 89 DA 18 **/
/** NEW CHIP PCM ID 20 DA 18 **/
ID89:
TS(3) OS(SO) OL(SO); /*OH(SO);*/
TS(3) OS(SO) OL(SO);
TS(3) OS(SO) OH(SO); /*OL(SO);*/
TS(3) OS(SO) OL(SO);
TS(3) OS(SO) OL(SO); /*OH(SO);*/
TS(3) OS(SO) OL(SO);
TS(3) OS(SO) OL(SO);
TS(3) OS(SO) OL(SO); /*OH(SO);*/

IDDA:
TS(3) OS(SO) OH(SO);
TS(3) OS(SO) OH(SO);
TS(3) OS(SO) OL(SO);
TS(3) OS(SO) OH(SO);
TS(3) OS(SO) OH(SO);
TS(3) OS(SO) OL(SO);
TS(3) OS(SO) OH(SO);
TS(3) OS(SO) OL(SO);

ID18:
TS(3) OS(SO) OL(SO);
TS(3) OS(SO) OL(SO);
TS(3) OS(SO) OL(SO);
TS(3) OS(SO) OH(SO);
TS(3) OS(SO) OH(SO);
TS(3) OS(SO) OL(SO);
TS(3) OS(SO) OL(SO);
TS(3) OS(SO) OL(SO);
OI(SO);
IH(CSN);;
ID(##) OI(##);
    END FAST;

    END BURST [
        WRITE ID=MESFILE 'FAILED TO READ ID %NL%';
        BRANCH U1_END;];

T4=TIME();
WRITE '%NL%** DEVICE ID TEST TIME USING CD = %2.3F% SECONDS%NL%'T4-T3 ;

```

```
/* Only execute Erase, Blank Check, and Program Steps if BE variable is set
to 1 */
```

```
IF BE=1 THEN[
```

```
/* Step 2 - Sector Erase 4Mb of data */
```

```
T3=TIME(); /* Record start time in variable T3 */
U1_4SE:
```

```
WRITE '%033%M** 4xSECTOR ERASE SERIAL FLASH. (4Mbits) %033:N';
```

```
BURST NAME='U1_ERASE' MAXTIME = 5;
START_MAIN:
```

```
FAST;
IC(CSN,SI) IH(CSN) IL(SI);;
```

```
SECTOR1:
```

```
/* WREN COMMAND 0000 0110*/
IC(SI) IL(CSN);
TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);
TS(2) IL(SI); TS(2) IH(SI); TS(2) IH(SI); TS(2) IL(SI);
IH(CSN);
```

```
/* SE COMMAND 1101 1000 */
IL(CSN);
TS(2) IH(SI); TS(2) IH(SI); TS(2) IL(SI); TS(2) IH(SI);
TS(2) IH(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);
```

```
/** 24 BITS SECTOR ADDRESS 0x0000 **/
```

```
HD(SI);
FLOOP(3) = 24 ;
TS(2) IC(SI) IL(SI);
END FLOOP;
IH(CSN);
```

```
GOSUB DATA_POLLING;
IH(CSN);
```

```
SECTOR2:
```

```
/* WREN COMMAND 0000 0110*/
IC(SI) IL(CSN);
TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);
TS(2) IL(SI); TS(2) IH(SI); TS(2) IH(SI); TS(2) IL(SI);
IH(CSN);
```

```
/* SE COMMAND 1101 1000 */
IL(CSN);
TS(2) IH(SI); TS(2) IH(SI); TS(2) IL(SI); TS(2) IH(SI);
TS(2) IH(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);
```

```
/** 24 BITS SECTOR ADDRESS 0x20000 **/
```

```
HD(SI);
FLOOP(3) = 6;
TS(2) IC(SI) IL(SI);
END FLOOP;
```

```

    HD(SI);
    FLOOP(3) = 1;
        TS(2) IC(SI) IH(SI);
    END FLOOP;
    HD(SI);
    FLOOP(3) = 17;
        TS(2) IC(SI) IL(SI);
    END FLOOP;

    IH(CSN);
    GOSUB DATA_POLLING;
    IH(CSN);

SECTOR3:
/* WREN COMMAND 0000 0110*/
IC(SI) IL(CSN);
TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);
TS(2) IL(SI); TS(2) IH(SI); TS(2) IH(SI); TS(2) IL(SI);
IH(CSN);

/* SE COMMAND 1101 1000 */
IL(CSN);
TS(2) IH(SI); TS(2) IH(SI); TS(2) IL(SI); TS(2) IH(SI);
TS(2) IH(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);

/** 24 BITS SECTOR ADDRESS 0x40000 **/
    HD(SI);
    FLOOP(3) = 5;
        TS(2) IC(SI) IL(SI);
    END FLOOP;
    HD(SI);
    FLOOP(3) = 1;
        TS(2) IC(SI) IH(SI);
    END FLOOP;
    HD(SI);
    FLOOP(3) = 18;
        TS(2) IC(SI) IL(SI);
    END FLOOP;

    IH(CSN);
    GOSUB DATA_POLLING;
    IH(CSN);

SECTOR4:
/* WREN COMMAND 0000 0110*/
IC(SI) IL(CSN);
TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);
TS(2) IL(SI); TS(2) IH(SI); TS(2) IH(SI); TS(2) IL(SI);
IH(CSN);

/* SE COMMAND 1101 1000 */
IL(CSN);
TS(2) IH(SI); TS(2) IH(SI); TS(2) IL(SI); TS(2) IH(SI);
TS(2) IH(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);

/** 24 BITS SECTOR ADDRESS 0x60000 **/
    HD(SI);

```

```

FLOOP(3) = 5;
    TS(2) IC(SI) IL(SI);
END FLOOP;
HD(SI);
FLOOP(3) = 2;
    TS(2) IC(SI) IH(SI);
END FLOOP;
HD(SI);
FLOOP(3) = 17;
    TS(2) IC(SI) IL(SI);
END FLOOP;

IH(CSN);
GOSUB DATA_POLLING;
IH(CSN);

ID(#) OI(#);
END FAST;

FASTSUB DATA_POLLING;
/* This Subroutine For Data Polling determines when Erase is complete */
/* RDSR COMMAND 0000 0101*/
$ IC(SI,CSN);
TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);
TS(2) IL(SI); TS(2) IH(SI); TS(2) IL(SI); TS(2) IH(SI);

/* check Write In Progress */
$ FLOOP(3)=10;
$ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3);
EXITIF PASSES OS(SO) OH(SO) TS(3);
$;$;$;$;$;
$ OI(SO) END FLOOP;
OS(SO) OH(SO); OI(SO) ;

/* SE ~500mSEC Delay Loop*/
$ FLOOP(3)=50000; /* ~50000 x 10uSEC ~ 500mSEC */
$; $; $; $; $; $; $; $; $; $; /* ~2uSEC*/
$; $; $; $; $; $; $; $; $; $; /* ~2uSEC*/
$; $; $; $; $; $; $; $; $; $; /* ~2uSEC*/
$; $; $; $; $; $; $; $; $; $; /* ~2uSEC*/
$; $; $; $; $; $; $; $; $; $; /* ~2uSEC*/
$ END FLOOP;

/* Exit Subroutine when status bit is low */
$ FLOOP(3)=64000;
$ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3);
$ TS(3); $ TS(3); $ TS(3);
EXITIF PASSES OS(SO) OL(SO) TS(3);
$;$;$;$;$;
$ OI(SO) END FLOOP;
OS(SO) OL(SO); OI(SO) ;
$ RETURN;
END FASTSUB DATA_POLLING;

END BURST [
    WRITE ID=MESFILE 'FAILED TO ERASED %NL%';
    BRANCH U1_END;];

```

```

T4=TIME();
WRITE '%NL%** 4xSECTOR ERASE TEST TIME USING D/S= %2.3F% SECONDS%NL%'T4-T3;

/* Step 3 - Blank Check test of 4Mb of data */

T3=TIME();
U1_BC4:
WRITE '%033%M** 4xSECTOR BLANK CHECK. (4Mbits)%033%:N';
BURST NAME='U1_4BC' MAXTIME = 5;

START_MAIN:
FAST;
IC(CSN,SI) IH(CSN) IL(SI);;

/* DOFR COMMAND 0011 1011*/
IL(CSN);
TS(2) IL(SI); TS(2) IL(SI); TS(2) IH(SI); TS(2) IH(SI);
TS(2) IH(SI); TS(2) IL(SI); TS(2) IH(SI); TS(2) IH(SI);

/* 24BITS ADDRESS */
HD(SI);
FLOOP=32; /* FAST READ IS 3 ADDRESS PLUS 1 DUMMY BYTES */
TS(2) IC(SI) IL(SI);
END FLOOP;

HD(SO,SI);

/* Read 4Mb of data using two pin Fast Read mode */
FLOOP=8192;
FLOOP=256;
TS(3) OS(SO,SI) OH(SO,SI)
END FLOOP;
END FLOOP;

IH(CSN);
ID(#) OI(#);
END FAST;
END BURST [
WRITE ID=MESFILE 'FAILED TO READ %NL%';
BRANCH U1_END;];

T4=TIME();
WRITE '%NL%** 4xSECTOR BLANK CHECK TIME USING D/S= %2.3F% SECONDS%NL%'T4-T3
;
]; /* END OF BE=1 */

```

```

/* Step 4 - Program 4Mb of data */

T3=TIME();

U1_DIFP:

/*****
/* 4Mb of programming data is stored in DSM in the Dataset PCMX2_DATASET */
/* Addresses to program are Imported into this Burst using Bitstring      */
/* variables LOAD_ADDR_BS_LO and LOAD_ADDR_BS_HI.                          */
*****/

BURST NAME='U1_WRITE' MAXTIME=30
DSMSTART='PCMX2_DATASET' DSM(297,289) DSMDOUBLE
IMPORT(289:LOAD_ADDR_TAB_LO:2048:LOAD_ADDR_BS_LO)
IMPORT(289:LOAD_ADDR_TAB_HI:160:LOAD_ADDR_BS_HI);

START_MAIN:

    FAST;
    IC(CSN,SI) IH(CSN) IL(SI);;;

SMEN:/* WREN COMMAND 0000 0110*/
IC(SI) IL(CSN); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);
TS(2) IL(SI); TS(2) IL(SI); TS(2) IH(SI); TS(2) IH(SI);
TS(2) IL(SI);
IH(CSN);

/* SMEN COMMAND CF 81 A23..A0 00 */
IC(SI) IL(CSN);
BY1:
TS(2) IH(SI); TS(2) IH(SI); TS(2) IL(SI); TS(2) IL(SI);
TS(2) IH(SI); TS(2) IH(SI); TS(2) IH(SI); TS(2) IH(SI);
BY2:
TS(2) IH(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);
TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IH(SI);
BY3_BY6:
FLOOP(3) = 32 ;
    TS(2) IL(SI);
END FLOOP;
IH(CSN);

/* RDSR COMMAND 0000 0101*/
IC(SI) IL(CSN);
TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);
TS(2) IL(SI); TS(2) IH(SI); TS(2) IL(SI); TS(2) IH(SI);

SMEN_POLL:
    /* check Write In Progress */
    $ FLOOP(3)=10;
    $ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3);
EXITIF PASSES OS(SO) OH(SO) TS(3);
    $;$;$;$;$;
    $ OI(SO) END FLOOP;
    OS(SO) OH(SO); OI(SO) ;

    /* polling program done */

```

```

    $ FLOOP(3)=10;
    $ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3);
    EXITIF PASSES OS(SO) OL(SO) TS(3);
    $; $; $; $; $;
    $ OI(SO) END FLOOP;
    OS(SO) OL(SO);

    TS(2) IH(CSN) OI(SO);;

/* HIGH MEMORY ADDRESS - SET ADDRESS - Loops through 8192 pages */
    LOAD DRIVE(1) LOAD_ADDR_TAB_HI;
    FLOOP(1) = 32;
    HD(SI);
    LOAD DRIVE(2) LOAD_ADDR_TAB_LO;
    FLOOP(2) = 256;

/* WREN COMMAND 0000 0110*/
    IC(SI) IL(CSN);
    TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);
    TS(2) IL(SI); TS(2) IH(SI); TS(2) IH(SI); TS(2) IL(SI);
    IH(CSN);

/* DIFP ON ALL ONES COMMAND 1101 0101 D5h */
    IC(SI) IL(CSN);
    TS(2) IH(SI); TS(2) IH(SI); TS(2) IL(SI); TS(2) IH(SI);
    TS(2) IL(SI); TS(2) IH(SI); TS(2) IL(SI); TS(2) IH(SI);

/** A23-A19 BITS ADDRESS**/
    HD(SI)
    FLOOP(3) = 5 ;
    TS(2) IC(SI) IL(SI);
    END FLOOP;

/** A18-A14 BITS ADDRESS**/
    HD(SI)
    FLOOP(3) = 5 ;
    TS(2) USE DRIVE(1)+; /* Set Hi Address */
    END FLOOP;

/** MOVE BACK 5 STEP FOR REUSE **/
    HD(SI)
    FLOOP(3) = 5 ;
    USE DRIVE(1)-;
    END FLOOP;

/** A13-A6 BITS ADDRESS**/
    HD(SI)
    FLOOP(3) = 8 ;
    TS(2) USE DRIVE(2)+ ; /* Set Lo Address */
    END FLOOP;

/** A6-A0 BITS ADDRESS AUTO INCREMENT **/
    HD(SI)
    FLOOP(3) = 6 ;
    TS(2) IC(SI) IL(SI);
    END FLOOP;

```

```

HD(SO,SI)
  FLOOP(3)=256;
  /* Program all 4Mb using data from DSM */
  TS(2) IE(SO,SI) DSMDATA+
  END FLOOP;

IH(CSN) ID(SO,SI);

/* RDSR COMMAND 0000 0101*/
IC(SI) IL(CSN);
TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);
TS(2) IL(SI); TS(2) IH(SI); TS(2) IL(SI); TS(2) IH(SI);

  /* check Write In Progress */
  $ FLOOP(3)=10;
  $ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3);
EXITIF PASSES OS(SO) OH(SO) TS(3);
  $;$;$;$;$;
  $ OI(SO) END FLOOP;
  OS(SO) OH(SO);

  /* 1.5 ~ 5ms PP program done */
  $ FLOOP(3)=1250;
  $ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3); $ TS(3);
EXITIF PASSES OS(SO) OL(SO) TS(3);
  $;$;$;$;$;
  $ OI(SO) END FLOOP;
  OS(SO) OL(SO);
TS(2) IH(CSN) OI(SO);

END FLOOP; /* FOR FLOOP2 */

/** MOVE FORWARD 5 STEP FOR WHEN LO ADDRESS FINISHED **/
HD(SI)
FLOOP(3) = 5 ;
  USE DRIVE(1)+;
END FLOOP;

END FLOOP; /* FOR FLOOP1 */

SMEX:
IC(CSN,SI) IH(CSN) IL(SI);;;

/* WRDI COMMAND 0000 0100*/
IL(CSN);
TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);
TS(2) IL(SI); TS(2) IH(SI); TS(2) IL(SI); TS(2) IL(SI);
IH(CSN);

/* WREN COMMAND 0000 0110*/
IL(CSN);
TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI); TS(2) IL(SI);
TS(2) IL(SI); TS(2) IH(SI); TS(2) IH(SI); TS(2) IL(SI);
IH(CSN);

```





```

TS(2) IH(SI);

/* 24BITS ADDRESS */
HD(SI);
FLOOP=32; /* FAST READ IS 3 ADDRESS PLUS 1 DUMMY BYTES */
TS(2) IC(SI) IL(SI);
END FLOOP ID(SI);

HD(SO,SI)

/* Fast Verify of 4Mb of Data using DSM */
FLOOP=8192;
FLOOP=256;
TS(3) OE(SO,SI) DSMDATA+
END FLOOP;
END FLOOP;

IH(CSN);
ID(#) OI(#);
END FAST;

END BURST [
WRITE ID=MESFILE 'FAILED TO READ %NL%';
BRANCH U1_END;];

T4=TIME();
WRITE '%NL%** VERIFY TEST TIME USING CDx2 OPT = %2.3F% SECONDS%NL%'T4-T3 ;
};

```

## Conclusion

The TestStation in-circuit test system is well equipped to handle the in-system programming challenges of the latest technology Phase Change Memory components.