MC68705 PANC Command Complete Specification

Detailed Analysis Based on Ghidra Disassembly

Introduction

The MC68705 Panel/Calendar Controller communicates with the Delilah DGA CPU using the PANC (Panel Control) protocol. This protocol uses a sophisticated command/response system where the DGA sends command bytes and receives response bytes through the IDB (Internal Data Bus) interface.

The communication is bidirectional and context-dependent, with the MC68705 acting as both a command processor (receiving instructions from DGA) and a data provider (sending time, status, and display information back to DGA).

Command Byte Format and Bit Definitions

Command Byte Structure (8 bits)

Bit	Name	Function	Values
7	**RTC**	RTC Operation Mode	0=Standard, 1=RTC Access
6	**MOD**	Processing Mode	0=Normal, 1=Enhanced
5	**RW**	RTC Read/Write	0=Read RTC, 1=Write RTC
4	**RSV**	Reserved	0=Normal, 1=Special
3	**DIR**	Data Direction	**0=MC68705→DGA (Write), 1=DGA→MC68705 (Read)**
2-0	**CMD**	Command Type	0-7 (8 possible commands)

Critical Bit Functions:

Bit 3 (DIR) - Data Direction Control

- **0**: MC68705 writes data TO DGA (output mode)
  - MC68705 reads RTC, formats data, sends to DGA via IDB
  - Function: `Read_MM58274_Time_Date_Output_To_DGA()`
- **1**: MC68705 reads data FROM DGA (input mode)
  - DGA sends command parameters to MC68705

- Function: Command switch processing (0-7)

Bit 2 (Command Bit 2)

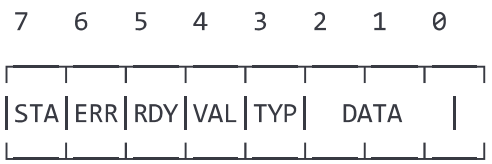
- Used as address/control bit in write operations
- Controls MM58274 register selection
- Must be set (=1) for RTC operations to proceed

Bit 5 (RW) - RTC Access Mode (when DIR=0)

- **0**: Standard data output mode
- **1**: RTC write/initialization mode

Response Byte Format

Response Byte Structure (8 bits)



Bit	Name	Function	Values
7	**STA**	Status Flag	0=Idle, 1=Active
6	**ERR**	Error Flag	0=OK, 1=Error
5	**RDY**	Ready Flag	0=Busy, 1=Ready
4	**VAL**	Data Valid	0=Invalid, 1=Valid
3	**TYP**	Data Type	0=Time, 1=Status
2-0	**DATA**	Data/Address	Time data or status info

Response Generation Functions:

- `Write_RegA_To_PortA_And_Latch_to_IDB(data)`: Standard response output
- `WriteDateBytesToPortA_LatchToIDB_7_0()`: Special response for address 0

Command Enumeration

c

```c
typedef enum {
    PANC_CMD_SET_DATETIME       = 0,  // Set complete time and date
    PANC_CMD_SET_TIME_ONLY      = 1,  // Set time only, preserve date
    PANC_CMD_READ_CPU_STATUS    = 2,  // Read system status (PONI/IONI)
    PANC_CMD_SET_DISPLAY_MODE   = 3,  // Set display mode/format
    PANC_CMD_LOAD_ROM_CHARS     = 4,  // Load display chars from ROM tables
    PANC_CMD_SET_SYSTEM_CONFIG  = 5,  // Set system configuration
    PANC_CMD_SET_DISPLAY_DIRECT = 6,  // Set direct display data
    PANC_CMD_SYSTEM_CONTROL     = 7   // System control and reset
} panc_command_t;

typedef enum {
    PANC_DIR_WRITE_TO_DGA  = 0,  // MC68705 → DGA (bit 3 = 0)
    PANC_DIR_READ_FROM_DGA = 1   // DGA → MC68705 (bit 3 = 1)
} panc_direction_t;

typedef enum {
    PANC_RTC_READ_MODE  = 0,  // Read from RTC (bit 5 = 0)
    PANC_RTC_WRITE_MODE = 1   // Write to RTC (bit 5 = 1)
} panc_rtc_mode_t;
```

Command Table

Enum	Value	Dir	Description	Input Bytes	Response	RTC Access
`PANC_CMD_SET_DATETIME`	0x08	R	Set complete time/date	5	None	No
`PANC_CMD_SET_TIME_ONLY`	0x09	R	Set time only	4	None	No
`PANC_CMD_READ_CPU_STATUS`	0x0A	R	Read CPU status	2	None	No
`PANC_CMD_SET_DISPLAY_MODE`	0x0B	R	Set display mode	3	None	No
`PANC_CMD_LOAD_ROM_CHARS`	0x0C	R	Load ROM characters	1	None	No
`PANC_CMD_SET_SYSTEM_CONFIG`	0x0D	R	Set configuration	1	None	No
`PANC_CMD_SET_DISPLAY_DIRECT`	0x0E	R	Set display data	4	None	No
`PANC_CMD_SYSTEM_CONTROL`	0x0F	R	System control	2	None	No
Write Operations	0x00-0x07	W	Output RTC data	0	1 byte	Yes

Legend:

- **Dir**: R=Read from DGA, W=Write to DGA

- **Input Bytes**: Number of data bytes following command
- **Response**: Data returned to DGA
- **RTC Access**: Whether command accesses MM58274

Command Byte Format Examples

Read Commands (DGA → MC68705)

```c
// Command 0: Set complete time and date
// Binary: 0000 1000 = 0x08
// Bits: RTC=0, MOD=0, RW=0, RSV=0, DIR=1, CMD=000
uint8_t cmd_set_datetime = (PANC_DIR_READ_FROM_DGA << 3) | PANC_CMD_SET_DATETIME;

// Command 7: System control with reset capability
// Binary: 0000 1111 = 0x0F
// Bits: RTC=0, MOD=0, RW=0, RSV=0, DIR=1, CMD=111
uint8_t cmd_system_ctrl = (PANC_DIR_READ_FROM_DGA << 3) | PANC_CMD_SYSTEM_CONTROL;
```

Write Commands (MC68705 → DGA)

```c
// Write RTC data for address 0 (standard read mode)
// Binary: 0000 0000 = 0x00
// Bits: RTC=0, MOD=0, RW=0, RSV=0, DIR=0, CMD=000
uint8_t write_addr0 = (PANC_DIR_WRITE_TO_DGA << 3) | 0x00;

// Write RTC data for address 3 (RTC write mode)
// Binary: 0010 0011 = 0x23
// Bits: RTC=0, MOD=0, RW=1, RSV=0, DIR=0, CMD=011
uint8_t write_addr3_rtc = (PANC_RTC_WRITE_MODE << 5) | (PANC_DIR_WRITE_TO_DGA << 3) | 0x03;
```

Command Processing Flow

Initial Command Processing

1. **Command byte received** via `Strobe_WMM_Read_IDB_Data()` → stored in `RAM_001c`
2. **Direction check**: `if ((RAM_001c & 8) == 0)` → bit 3 = 0 means Write to DGA
3. **If Write to DGA**: Execute `Read_MM58274_Time_Date_Output_To_DGA()`
4. **If Read from DGA**: Execute command switch `(RAM_001c & 7)`

Write to DGA Operations (Bit 3 = 0)

When command bit 3 = 0, the MC68705 outputs data to the DGA CPU.

Function: Read_MM58274_Time_Date_Output_To_DGA()

Input: Command byte in `RAM_001c`

Processing:

1. **Extract address**: `RAM_001c = RAM_001c & 7` (bits 2-0)
2. **Set MM58274 address**: `PORTC = PORTC & 0xf8 | RAM_001c`
3. **Check bit 2**: `if ((bVar1 & 4) == 0) return` - early exit if bit 2 clear
4. **Clear control bit**: `PORTB = PORTB & 0xdf` (clear PB5)
5. **Invert address**: `RAM_001c = bVar1 & 3 ^ 3` (XOR with 3)

Two operation modes based on bit 5:

Mode 1: Bit 5 = 0 (Standard Read)

```
c

PORTB = bVar3 & 0x9f;  // Clear PB6 and PB5
uVar2 = Strobe_WMM_Read_IDB_Data();  // Read additional data
*(uint8_t*)(RAM_001c + 0x47) = uVar2;  // Store at 0x47+address
Write_RegA_To_PortA_And_Latch_to_IDB();  // Output to DGA
PORTB = PORTB | 0x20;  // Set PB5

if (RAM_001c == 0) {  // Special case for address 0
    CopyData_In_Ram_Dest_0x46_0x4e();  // Complex time calculation
    WriteDateBytesToPortA_LatchToIDB_7_0();  // Output result
}
```

Mode 2: Bit 5 = 1 (RTC Write Mode)

```c
PORTB = PORTB | 0x40;  // Set PB6 (RTC mode)
Strobe_WMM_Read_IDB_Data();  // Read data (discarded)

if (RAM_001c == 3) {  // Special case for address 3
    Initialize_MM58274_RTC_Chip();  // Reinitialize RTC
    Calc_Something_Dest_0x48_0x4f();  // Calculate display format
}

Write_RegA_To_PortA_And_Latch_to_IDB(*(uint8_t*)(RAM_001c + 0x47));  // Output stored data
PORTB = PORTB | 0x20;  // Set PB5
```

Output Functions:

- **Write_RegA_To_PortA_And_Latch_to_IDB()**: `PORTA = data; PORTB &= 0xfe; PORTB |= 1; DDRA = 0;`
- **WriteDateBytesToPortA_LatchToIDB_7_0()**: `PORTB &= 0xfd; PORTA = 1; PORTB |= 2; DDRA = 0;`

Read from DGA Operations (Bit 3 = 1)

When command bit 3 = 1, the MC68705 processes commands from the DGA CPU.

Command 0: Set Complete Time and Date

Command Byte: `0x08` (bit 3=1, bits 2-0=000)

Input Sequence: 5 bytes from DGA

```c
RAM_0014 = Strobe_WMM_Read_IDB_Data();  // Byte 1 → 0x14
RAM_0016 = Strobe_WMM_Read_IDB_Data();  // Byte 2 → 0x16
RAM_0015 = Strobe_WMM_Read_IDB_Data();  // Byte 3 → 0x15
RAM_0018 = Strobe_WMM_Read_IDB_Data();  // Byte 4 → 0x18
RAM_0017 = Strobe_WMM_Read_IDB_Data();  // Byte 5 → 0x17
```

Internal Processing:

1. **Clear control flags**: `RAM_001b = RAM_001b & 0xe5` (clear bits 4,3,1)
2. **Set display character 0**: `RAM_0010 = 0x50` ('P')
3. **Mode selection based on RAM_001a bit 2**:

Normal Mode (RAM_001a & 4 == 0):

c

```
RAM_0011 = 0x45;  // 'E'
RAM_0012 = 0x58;  // 'X'
RAM_0013 = 0x4d;  // 'M'
// Display shows "PEXM"
```

Test Mode (RAM_001a & 4 != 0):

c

```
RAM_0011 = 0x54;  // 'T'
BYTE_0052 = (RAM_001a & 0x18) >> 1;  // Extract bits 4-3, shift right 1
RAM_0013 = RAM_001a & 3 | BYTE_0052;  // Combine bits 1-0 with shifted bits
if (RAM_0013 < 8) {
    RAM_0013 = RAM_0013 + 0x30;  // Convert to ASCII digit
    RAM_0012 = 0;
} else {
    RAM_0012 = 0;
}
// Display shows "PT##" where ## is calculated digit
```

Memory Map:

- `RAM_0014`: Time/Date Byte 1
- `RAM_0015`: Time/Date Byte 3
- `RAM_0016`: Time/Date Byte 2
- `RAM_0017`: Time/Date Byte 5
- `RAM_0018`: Time/Date Byte 4

Response: None (write operation)

Command 1: Set Time Only

Command Byte: `0x09` (bit 3=1, bits 2-0=001)

Input Sequence: 4 bytes from DGA

c

```c
RAM_0018 = Strobe_WMM_Read_IDB_Data();  // Byte 1 → 0x18
RAM_0017 = Strobe_WMM_Read_IDB_Data();  // Byte 2 → 0x17
RAM_0016 = Strobe_WMM_Read_IDB_Data();  // Byte 3 → 0x16
RAM_0015 = Strobe_WMM_Read_IDB_Data();  // Byte 4 → 0x15
RAM_0014 = 0;  // Clear byte 1
```

Internal Processing:

1. **Update control flags**: `RAM_001b = RAM_001b & 0xf5 | 0x10` (clear bit 3, set bit 4)

Memory Map:

- `RAM_0014`: Cleared to 0
- `RAM_0015`: Time Byte 4
- `RAM_0016`: Time Byte 3
- `RAM_0017`: Time Byte 2
- `RAM_0018`: Time Byte 1

Response: None (write operation)

Command 2: Read System Status

Command Byte: `0x0A` (bit 3=1, bits 2-0=010)

Input Sequence: 2 bytes from DGA

c

```c
RAM_0018 = Strobe_WMM_Read_IDB_Data();  // Parameter 1 → 0x18
RAM_0017 = Strobe_WMM_Read_IDB_Data();  // Parameter 2 → 0x17
```

Internal Processing:

1. **Update control flags**: `RAM_001b = RAM_001b & 0xe7 | 2` (clear bits 4,3, set bit 1)
2. **Set parameters**: `RAM_001d = 0x10; RAM_001e = 0;`
3. **Read CPU status from Port D and format for display**:

c

```c
RAM_0012 = (CPU_Status_Ring_Level_PONI_IONI & 0xf) + 0x10;    // Lower 4 bits + 0x10
RAM_0013 = ((CPU_Status_Ring_Level_PONI_IONI & 0x30) >> 4) + 0x3a;  // Bits 5-4 >> 4 + 0x3A
```

Status Interpretation:

- **CPU_Status_Ring_Level_PONI_IONI**: Current CPU status from Port D
- **Lower 4 bits**: Ring level and status flags
- **Bits 5-4**: PONI/IONI status
- **Display characters**: Formatted as ASCII values for panel display

Memory Map:

- `RAM_0012`: Status display character (lower nibble + 0x10)
- `RAM_0013`: Status display character (upper bits + 0x3A)
- `RAM_0017`: Parameter 2
- `RAM_0018`: Parameter 1

Response: None (write operation)

Command 3: Set Display Mode

Command Byte: `0x0B` (bit 3=1, bits 2-0=011)

Input Sequence: 3 bytes from DGA

```c
RAM_0016 = Strobe_WMM_Read_IDB_Data();  // Byte 1 → 0x16
RAM_0018 = Strobe_WMM_Read_IDB_Data();  // Byte 2 → 0x18
RAM_0017 = Strobe_WMM_Read_IDB_Data();  // Byte 3 → 0x17
```

Internal Processing:

1. **Update control flags**: `RAM_001b = RAM_001b & 0xed | 8` (clear bit 4, set bit 3)
2. **Clear specific registers**: `RAM_0015 = 0; RAM_0014 = 0;`

Memory Map:

- `RAM_0014`: Cleared to 0
- `RAM_0015`: Cleared to 0
- `RAM_0016`: Mode Byte 1
- `RAM_0017`: Mode Byte 3
- `RAM_0018`: Mode Byte 2

Response: None (write operation)

Command 4: Load Display Characters from ROM

Command Byte: `0x0C` (bit 3=1, bits 2-0=100)

Input Sequence: 1 byte from DGA

```
c

BYTE_0052 = Strobe_WMM_Read_IDB_Data();  // Character selection index
```

Internal Processing:

1. **Table 1 lookup (ROM address 0x0D12)**:

```
c

bVar1 = (BYTE_0052 & 3) * 2;  // Use bits 1-0, multiply by 2
RAM_0013 = ROM_TABLE_0D12[bVar1];      // Load character 3
RAM_0012 = ROM_TABLE_0D12[bVar1 + 1];  // Load character 2
```

2. **Table 2 lookup (ROM address 0x0D1A)**:

```
c

bVar1 = (BYTE_0052 & 0x18) >> 2;  // Use bits 4-3, shift right 2
RAM_0011 = ROM_TABLE_0D1A[bVar1];      // Load character 1
RAM_0010 = ROM_TABLE_0D1A[bVar1 + 1];  // Load character 0
```

Character Loading Logic:

- **Bits 1-0**: Select character pair from Table 1 (0x0D12)
- **Bits 4-3**: Select character pair from Table 2 (0x0D1A)
- **Each table entry**: 2 bytes per character pair
- **Result**: 4 display characters loaded into RAM_0010-0013

Memory Map:

- `RAM_0010`: Character 0 (from Table 2)
- `RAM_0011`: Character 1 (from Table 2)
- `RAM_0012`: Character 2 (from Table 1)
- `RAM_0013`: Character 3 (from Table 1)
- `BYTE_0052`: Character selection index

Response: None (write operation)

Command 5: Set System Configuration

Command Byte: `0x0D` (bit 3=1, bits 2-0=101)

Input Sequence: 1 byte from DGA

```c
RAM_001a = Strobe_WMM_Read_IDB_Data();  // Configuration value → 0x1A
```

Internal Processing:

- **Direct storage**: Configuration byte stored in `RAM_001a`
- **No additional processing**: Value used by other commands (e.g., Command 0 mode selection)

Memory Map:

- `RAM_001a`: System configuration flags

Configuration Bits (based on usage in Command 0):

- **Bit 2**: Mode select (0=Normal "PEXM", 1=Test "PT##")
- **Bits 4-3**: Test mode format selection
- **Bits 1-0**: Test mode options

Response: None (write operation)

Command 6: Set Direct Display Data

Command Byte: `0x0E` (bit 3=1, bits 2-0=110)

Input Sequence: 4 bytes from DGA

```c
RAM_0011 = Strobe_WMM_Read_IDB_Data();  // Character 1 → 0x11
RAM_0010 = Strobe_WMM_Read_IDB_Data();  // Character 0 → 0x10
RAM_0013 = Strobe_WMM_Read_IDB_Data();  // Character 3 → 0x13
RAM_0012 = Strobe_WMM_Read_IDB_Data();  // Character 2 → 0x12
```

Internal Processing:

- **Direct loading**: Display characters loaded directly into display buffer
- **No transformation**: Characters used as-is for panel display

Memory Map:

- `RAM_0010`: Display character 0
- `RAM_0011`: Display character 1
- `RAM_0012`: Display character 2
- `RAM_0013`: Display character 3

Response: None (write operation)

Command 7: System Control and Reset

Command Byte: `0x0F` (bit 3=1, bits 2-0=111)

Input Sequence: 2 bytes from DGA

```c
RAM_001d = Strobe_WMM_Read_IDB_Data();  // Parameter 1 → 0x1D
RAM_001e = Strobe_WMM_Read_IDB_Data();  // Parameter 2 → 0x1E
```

Internal Processing:

Soft Reset Check:

```c
if ((RAM_001e & 0xf0) == 0x40) {  // Upper nibble = 0x4
    MR_Misc_register = 0x40;
    TCR_Timer_Control_Register = 0x47;
    PCR_Program_Control_Register = 1;
    DDRA = 0;
    DDRB = 0;
    DDRC = 0;
    RESET();  // Jump to reset vector - DOES NOT RETURN
}
```

Feature Control (if not reset):

```c
RAM_001b = RAM_001b & 0xfb;  // Clear bit 2

if ((RAM_001e & 0xf0) == 0x10) {  // Upper nibble = 0x1
    RAM_001b = RAM_001b | 4;  // Set bit 2
}
```

Parameter Processing:

c

```c
// Complex bit manipulation for utilization parameters
RAM_001e = ((RAM_001e & 0xf) << 1 | RAM_001d >> 7) << 1 | (byte)(RAM_001d << 1) >> 7;
RAM_001d = RAM_001d & 0x3f;  // Keep lower 6 bits
```

Control Logic:

- **0x4X**: Immediate soft reset (system restart)
- **0x1X**: Enable feature (set control bit 2)
- **Other**: Clear feature flag, process parameters

Memory Map:

- `RAM_001d`: Processed parameter 1 (masked to 6 bits)
- `RAM_001e`: Processed parameter 2 (bit-shifted result)
- Control flags in `RAM_001b` bit 2

Response: None (write operation), or system reset

Post-Command Processing

After all read commands (0-7), the following operations are executed:

Conditional CPU Utilization Processing:

c

```c
if ((RAM_001d & 1) != 0) {  // If parameter 1 bit 0 set
    Calculate_CPU_Ring_Utilization();
}
```

Statistics Processing (executed twice):

c

```c
Process_CPU_Utilization_Statistics();
Process_CPU_Utilization_Statistics();
```

Memory Layout Summary

Display Characters (0x10-0x13):

- `0x10`: Display character 0

- `0x11`: Display character 1
- `0x12`: Display character 2
- `0x13`: Display character 3

Time/Date Data (0x14-0x18):

- `0x14`: Time/Date byte 1 (Command 0,1)
- `0x15`: Time/Date byte 3 (Command 0,1)
- `0x16`: Time/Date byte 2 (Command 0,1,3)
- `0x17`: Time/Date byte 5 (Command 0,1,2,3)
- `0x18`: Time/Date byte 4 (Command 0,1,2,3)

System Configuration (0x1A-0x1E):

- `0x1A`: System configuration (Command 5)
- `0x1B`: Control flags (modified by Commands 0,1,2,3,7)
- `0x1C`: Last command byte
- `0x1D`: Utilization parameter 1 (Command 2,7)
- `0x1E`: Utilization parameter 2 (Command 2,7)

MM58274 RTC Buffer (0x20-0x26):

- `0x20`: MM58274 tenths seconds
- `0x21`: MM58274 units seconds
- `0x22`: MM58274 tens seconds
- `0x23`: MM58274 units minutes
- `0x24`: MM58274 tens minutes
- `0x25`: MM58274 units hours
- `0x26`: MM58274 tens hours

Working Buffers:

- `0x47-0x4A`: Data storage buffer (Write operations)
- `0x52`: Character selection index (Command 4)

Command Summary Table

Command	Direction	Inputs	Memory Modified	Special Processing
0	Read	5 bytes	0x10-0x18, 0x1B	Mode-dependent display ("PEXM"/"PT##")
1	Read	4 bytes	0x15-0x18, 0x1B	Time-only mode, clear 0x14
2	Read	2 bytes	0x12-0x13, 0x17-0x18, 0x1B, 0x1D-0x1E	CPU status formatting
3	Read	3 bytes	0x16-0x18, 0x1B	Display mode, clear 0x14-0x15
4	Read	1 byte	0x10-0x13, 0x52	ROM table lookup
5	Read	1 byte	0x1A	Direct configuration storage
6	Read	4 bytes	0x10-0x13	Direct display character loading
7	Read	2 bytes	0x1B, 0x1D-0x1E	Reset or feature control
Write	Write	0 bytes	Various	MM58274 time output to DGA

All commands except Write operations trigger CPU utilization processing if `RAM_001d` bit 0 is set.