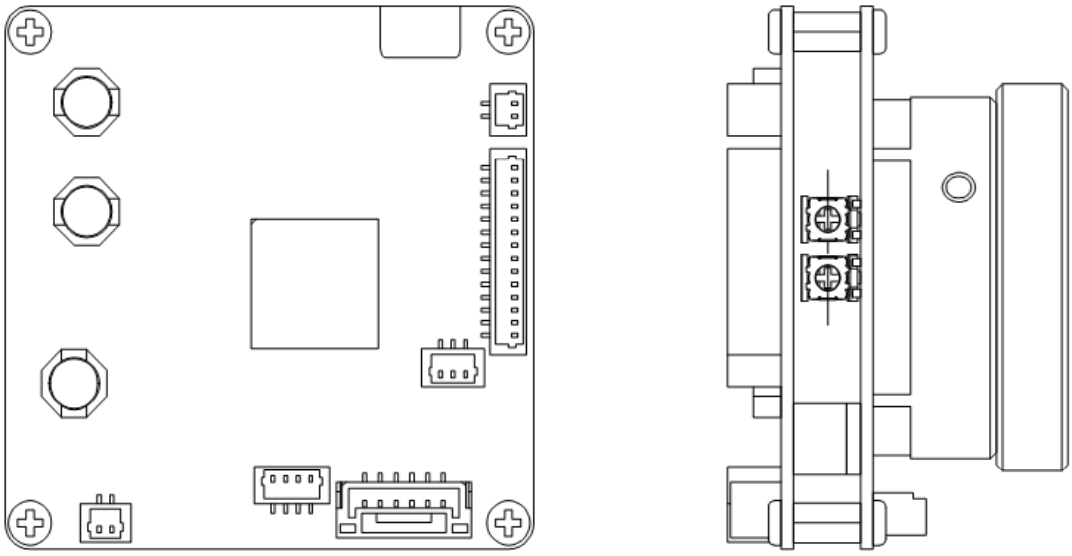
5.2 SPI COMMUNICATION

By using the SPI communication connector (J7), various functions of WAT-910BD can be controlled directly, without using OSD.

(1) SPI communication connector

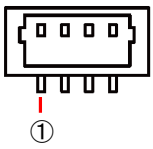
Refer to the following figure for the position of SPI communication connector (J7).



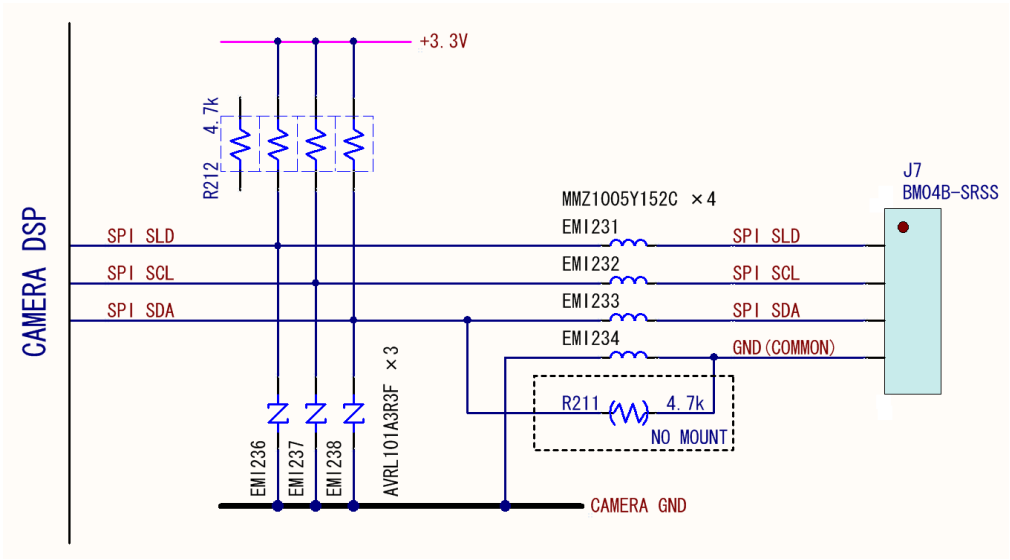
(Fig.5.2-1) SPI Communication Connector(J7)

5.2.1 ELECTRICAL CHARACTERISTICS

The pin arrangement and the outline circuit diagram of each pin are shown below.

(Fig.5.2-2) Pin Function and Descriptions							
	Parts Shape	Name	Parts No.	manufacturer	Pin No.	Description	I/O
J7		SPI Communication	BM04B-SRSS	JST	①	SPI SLD (active L)	I
					②	SPI SCL	I
					③	SPI SDA	I/O
					④	GND (common)	GND

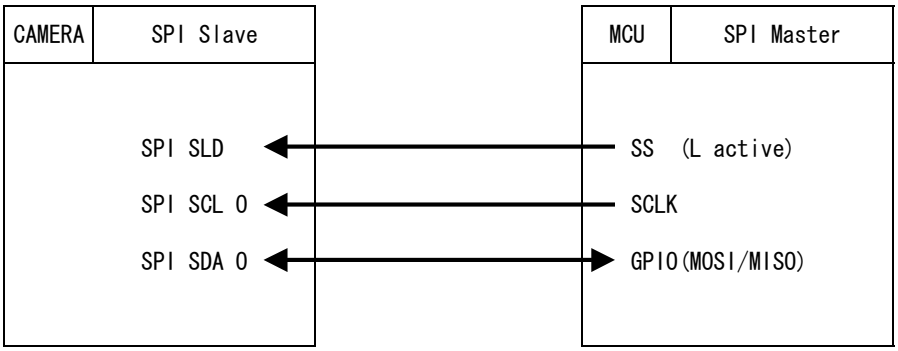
All pins of J7 connector are directly wired to Camera DSP through the bead(EMI231-234, inductor for high frequency).
(3.3[V], CMOS logic input/output. There are no buffers/drivers.)
And the varistor elements are connected between each terminal and camera GND as countermeasure for the electrostatic discharge and/or surge damage.
Each pin is connected to 3.3V(internal camera circuit) through resistors (4.7k ohm).



(Fig.5.2-3) partial schematic of SPI Communication

(1) Preparation

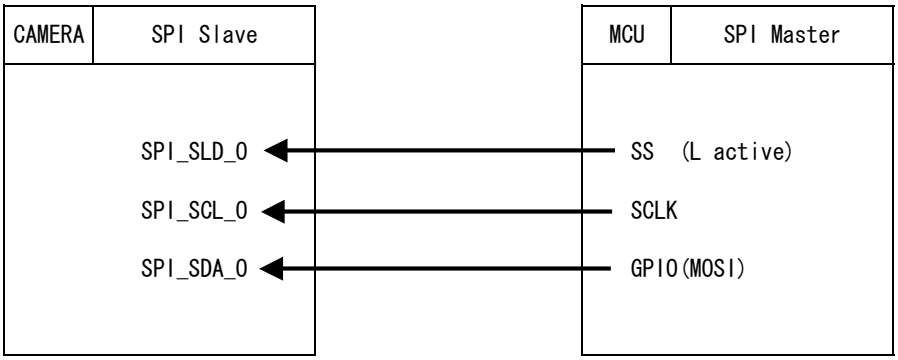
Though the communication specification of WAT-910BD applies to SPI bus specification fundamentally, it is implemented as 3 wire type.
Connection of the communication port of external MCU (master) and a camera (slave) is made as follows.



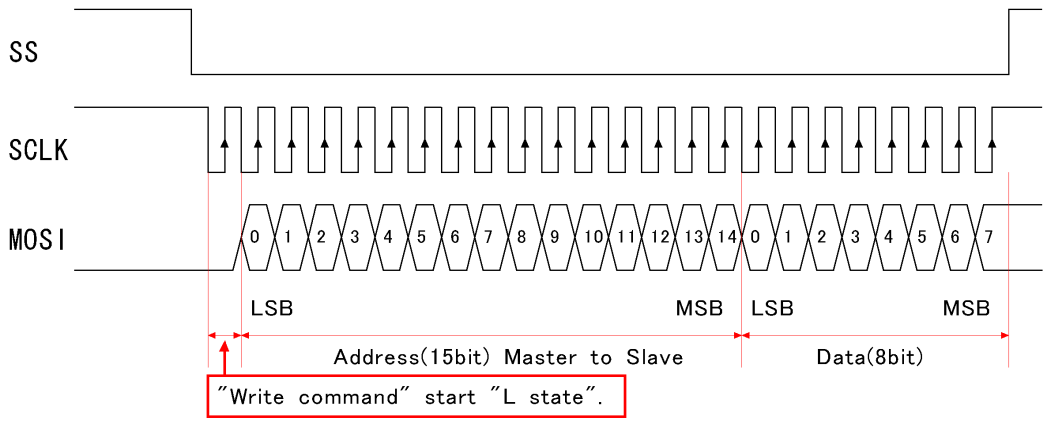
Both directions (input and output) are required for GPIO of external MCU (master), and in order to do bidirectional communication (MISO/MOSI), it needs to be switched direction (input or output).

(2) Command and Timing Chart(SPI Write Command)

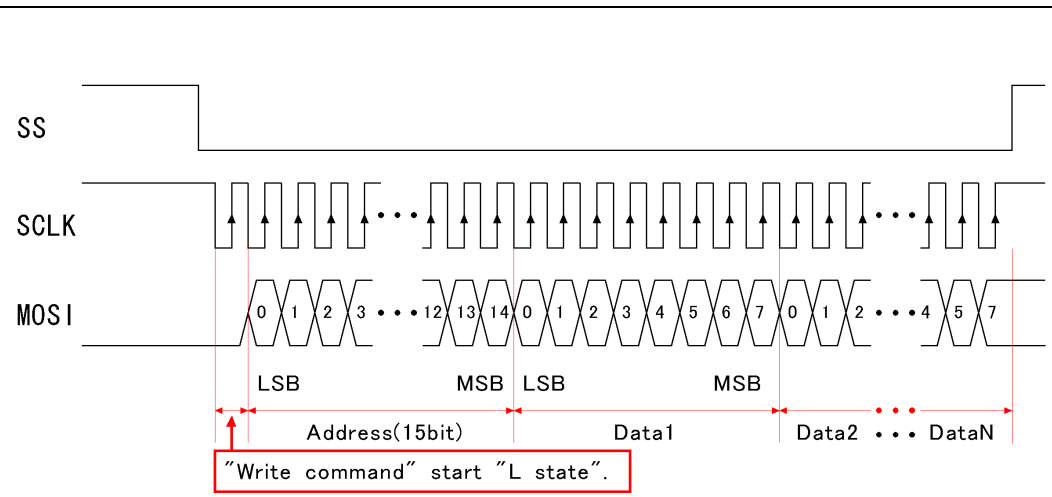
All the Write commands are direction from the external MCU to the camera DSP (master to slave). So, GPIO of external MCU is used as MOSI and a command is outputted to camera SPI SDA as it is. (GPIO of external MCU is always used as output port.)



H/L level of MISO/MOSI changes are synchronize with SCLK falling edge, and Data latch is done synchronizing with the rising up edge of SCLK.
This operation is the same about both directions(master to slave and slave to master communication).
The timing charts of Write command (master to slave only) are shown with following.



(Fig.5.2-4) 1 byte Write command (master to slave timing chart)



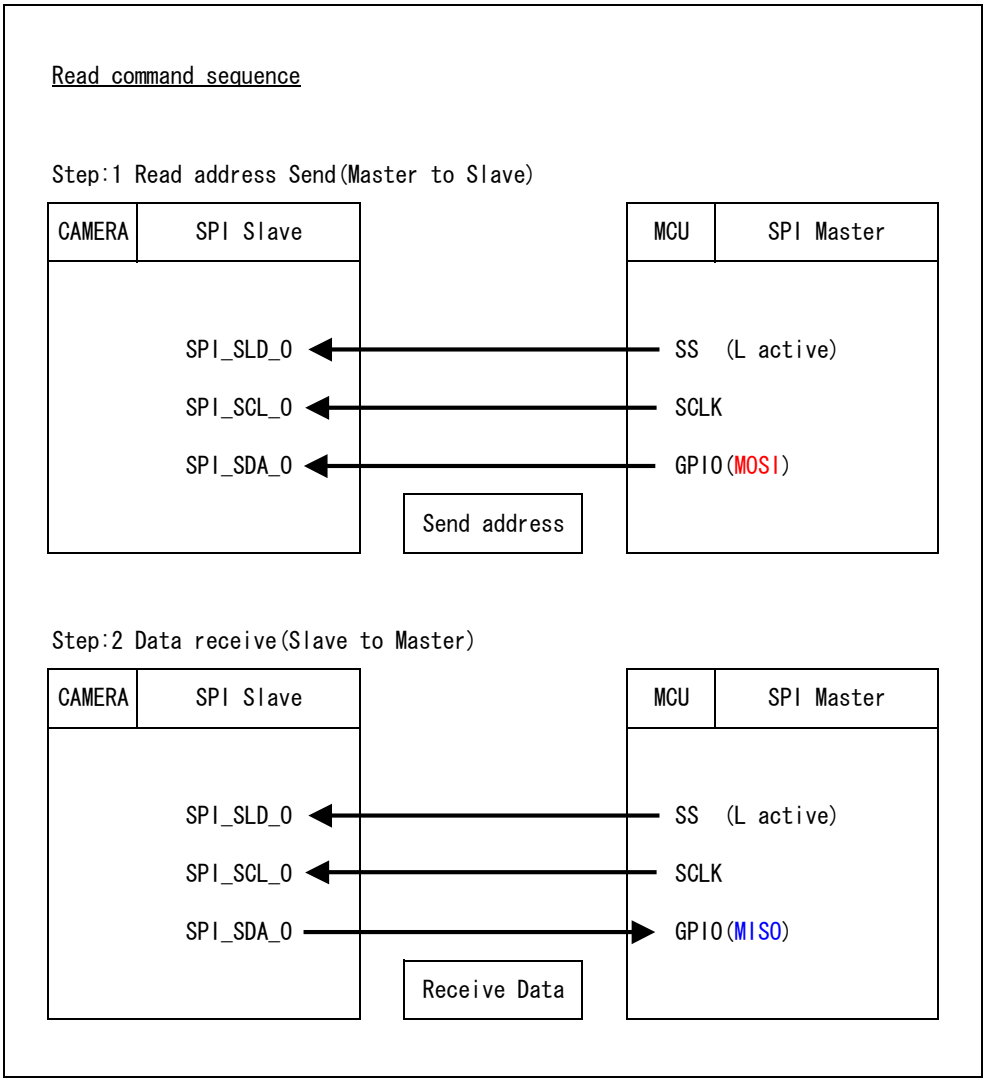
(Fig.5.2-5) N byte Write command (master to slave timing chart)

(3) Command and Timing Chart(SPI Read Command)

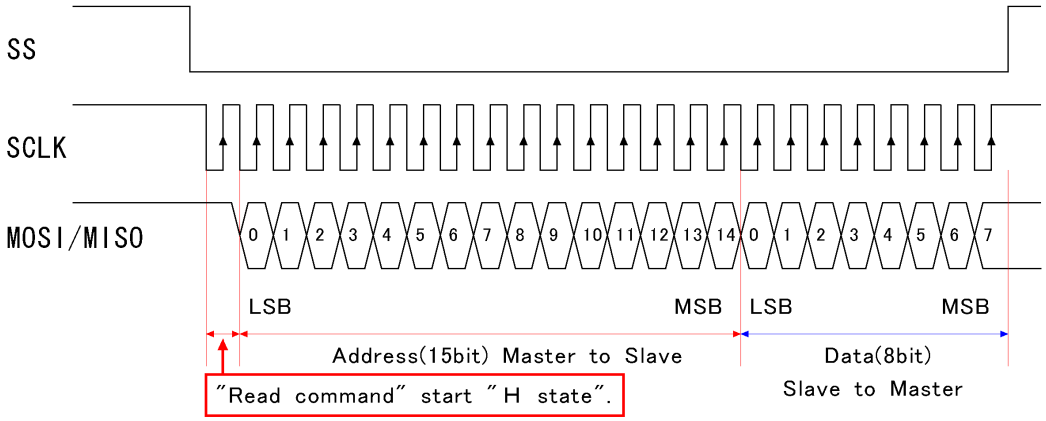
In the case of Read command, after sending the address of the needed data from external MCU to Camera DSP (master -> slave), it is outputted to external MCU from Camera DSP (slave -> master).

So, GPIO of external MCU is used as MOSI at the time of address sending, and it is used as MISO at the time of data receive.

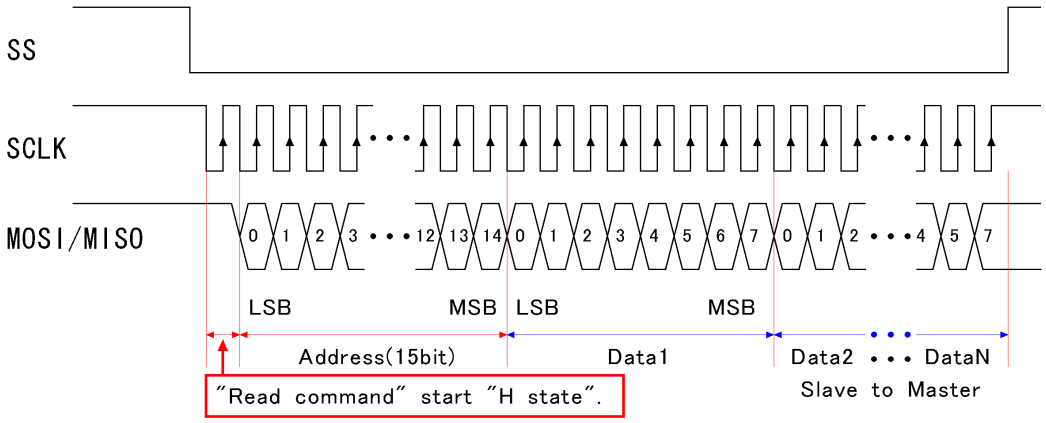
(At first, when sending address period, GPIO of external MCU is used as output port, then it is switched to input port.)



The timing charts of Read command (after send command, it will be returned response to master from slave) is shown with following.

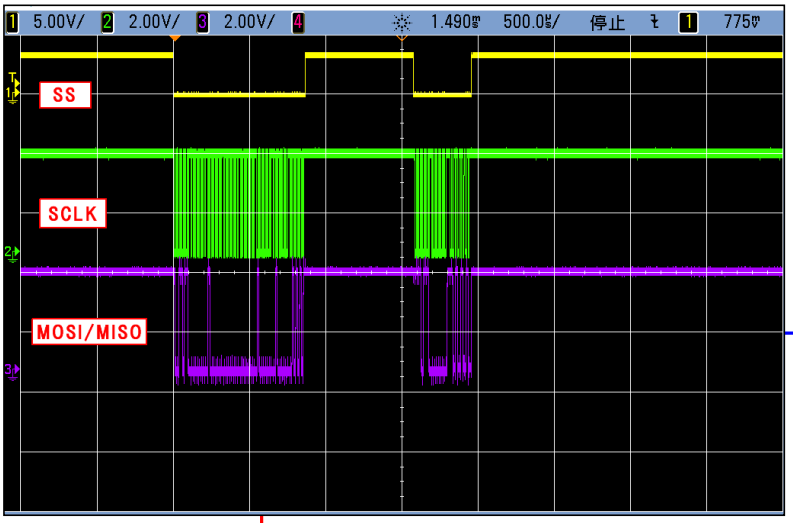


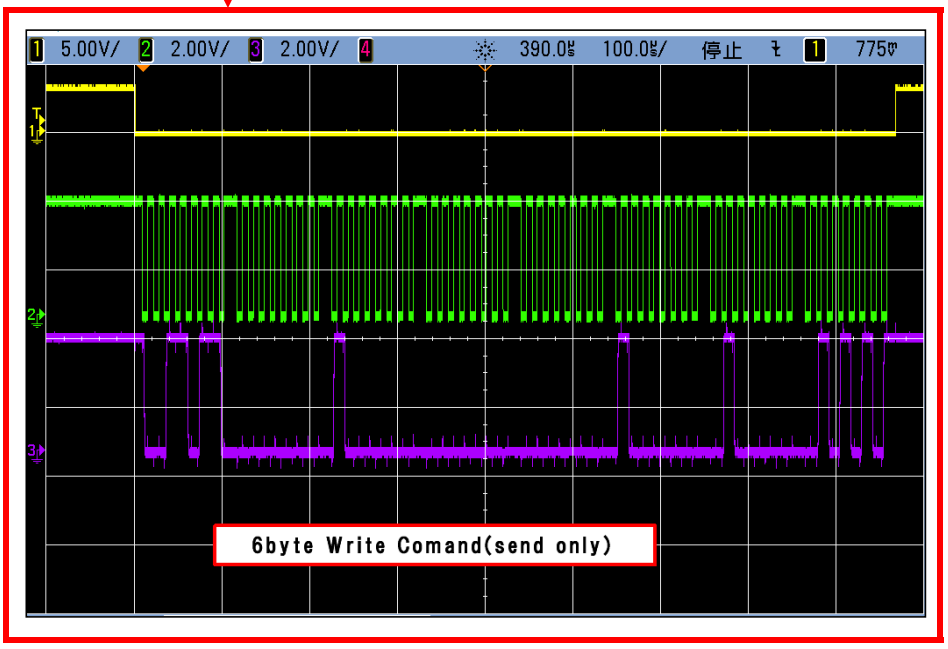
(Fig.5.2-6) 1 byte Read command (master to slave and slave to master timing chart)

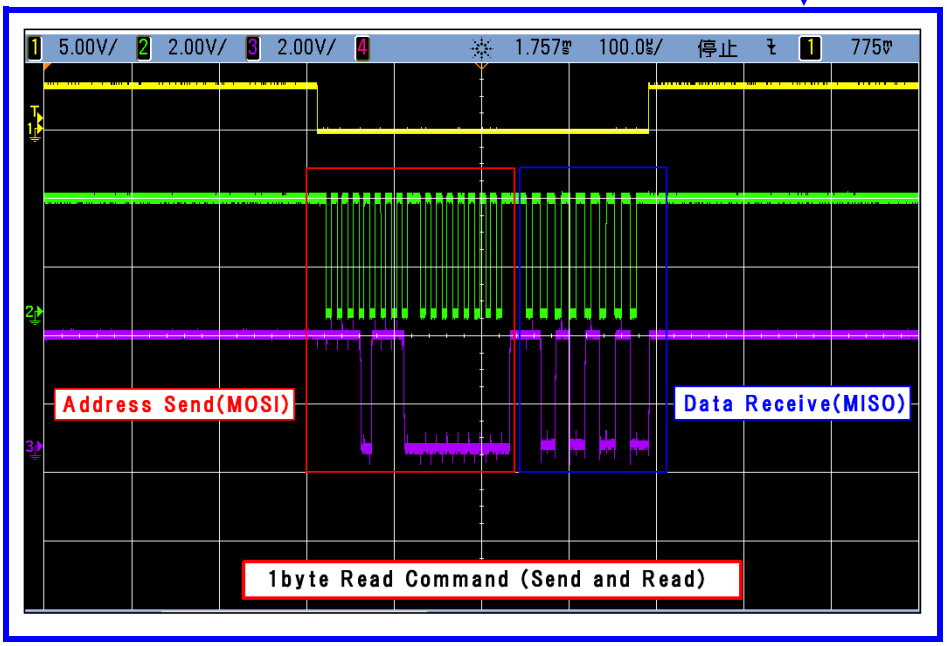


(Fig.5.2-7) N byte Read command (master to slave and slave to master timing chart)

(4) Actual wave form of "SPI Read Command"







5.2.2 CTL-COMMAND PROTOCOL

Camera CTL-COMMAND is executed by inside MCU on camera DSP.
"CTL-COMMAND processing" is doing by "indirect parameter accessing".
Camera MCU check the indirect registers to fetch CTL-COMMAND, and execute it immediately.
Thus, to control camera and/or to read/write camera parameter, external MCU should send command, address, parameter value, check-sum and status CODE to indirect registers by using SPI write method.
Indirect registers to use SPI communications are shown in following table.

(Fig.5.2-8) Indirect Registers for SPI Control

No.	Symbol	Address	Description
1	C1	0x0036	Command1
2	C2	0x0037	Command2
3	ADR	0x0038	Parameter Address
4	DAT	0x0039	Parameter Data
5	CS	0x003A	Check sum (C1+C2+ADR+DAT)
6	ST	0x003B	status CODE

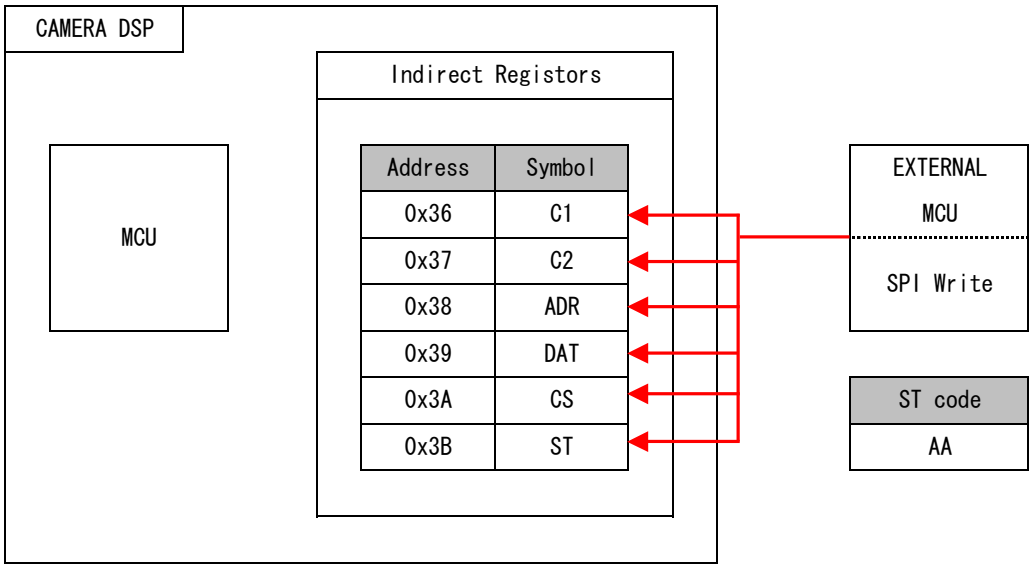
The value of ST(0x3B) register shows the status of the end of normal execution / error of CTL-COMMAND, and turns into one of the following values.

(Fig.5.2-9) ST register CODE

No.	CODE	Description
1	AA	DO COMMAND (This status code is written by EXT.MCU only.)
2	55	EXEC.NORMALLY (This status code is written by CAM MCU only.)
3	A5	EXEC.ERROR (This status code is written by CAM MCU only.)

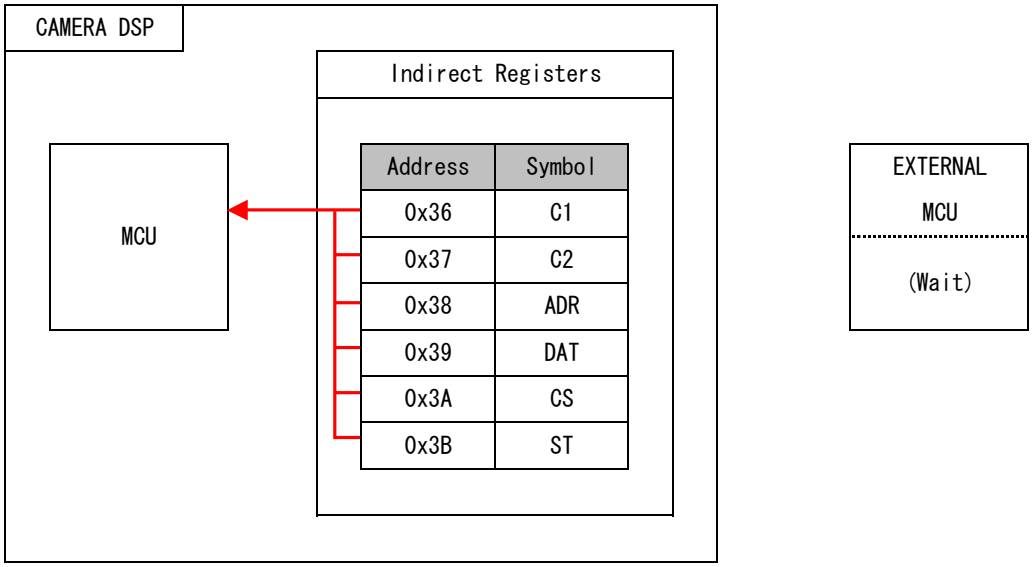
Execution of camera CTL-COMMAND is specifically expressed at the following steps.

Step.1: CTL-COMMAND SEND(EXT.MCU -> CAM DSP)



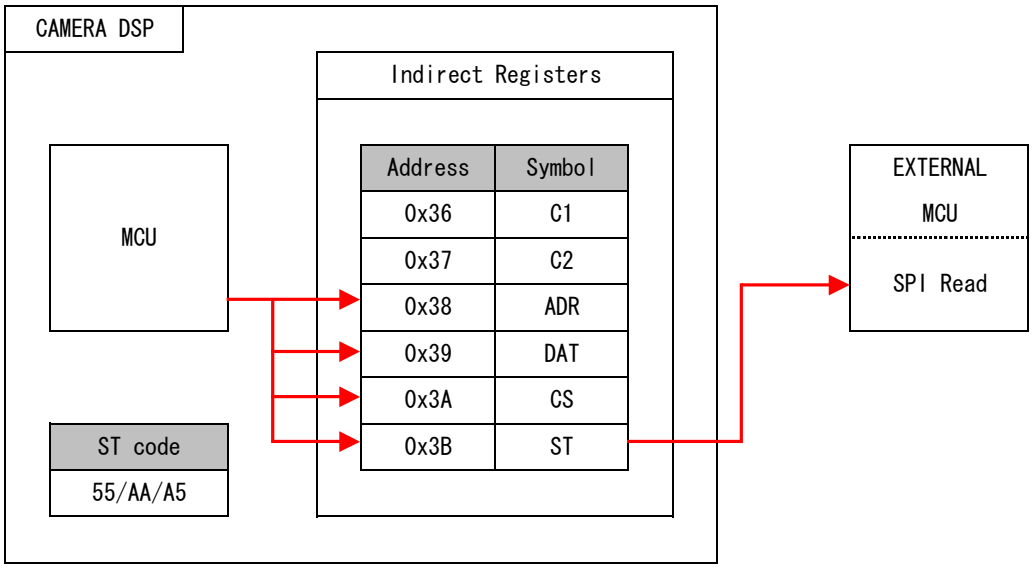
Set CTL-COMMAND to indirect registers from external MCU by SPI write command.

Step.2: COMMAND LATCH and EXECUTION(CAM MCU)



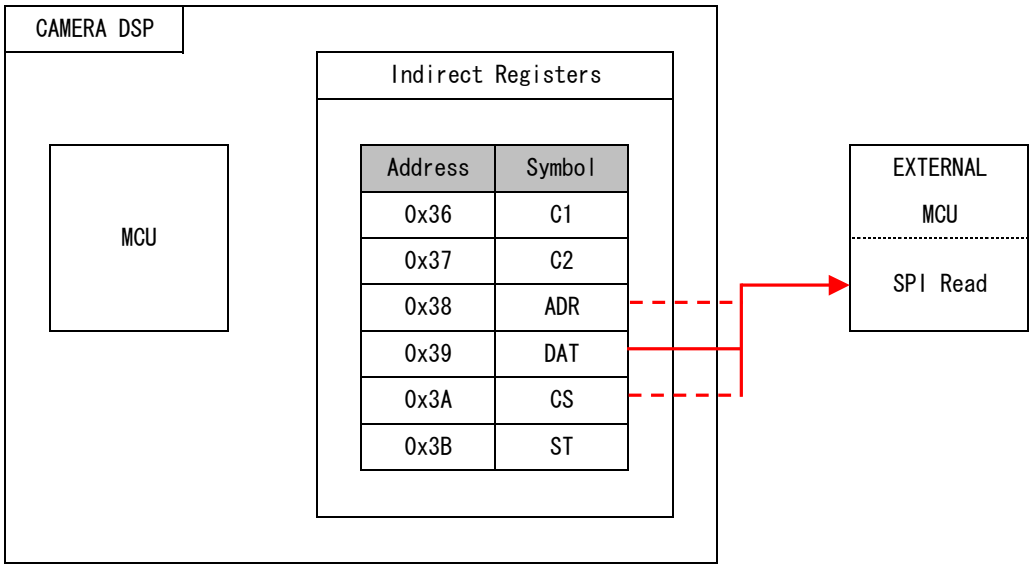
CAM MCU capture CTL-COMMAND from indirect registers and execute them.

Step.3: CHECK ST CODE



After command execution, camera MCU set execution results in indirect registers.
For checking the end of CTL-COMMAN execution , external MCU should do SPI read and check data in ST(0x3B) register.
If ST code is 55, it means "Command Execution Normally END" of CTL-COMMAND.
In this case, CAM MCU response is set ADR/DAT/CS(0x38/0x39/0x3A) registers.
If ST code is A5, it means "Command Execution ERROR" of CTL-COMMAND.
In this case, we should check command format(correctness of ADR/DATA/CS data), and send correct command to indirect registers.
If ST code is stay AA, CTL-COMMAND execution is in execution.

Step.4: RESPONSE READ



After checking ST (CODE:55) by using SPI Read command, get DAT(0x39) by using SPI read command.
This is a basic step for CTL-COMMAND(camera parameter read).
If you need fully secure, it need to check indirect register value from ADR to CS (from 0x38 to 0x3A).
In this case, you get CS value which equal ADR+DAT value.

5.2.3 CAMERA CTL-COMMAND

"Camera CTL-COMMAND" are shown in below.

(1) CAMERA PARAMETER READ1(parameter address: 0x400 - 0x4FF)

SPI Write "data set"

Symbol	C1	C2	ADR	DAT	CS	ST
Value	00	00	*1	00	*2	AA

*1: Parameter address lower byte. Assignable value range is 00 to FF.
*2: Check sum (lower byte of "C1 + C2 + ADR + DAT)
*In READ command, DAT value is always set 00.

SPI Read "data set"(camera MCU response)

Symbol	ADR	DAT	CS	ST
Value	*1	*2	*3	*4

*1: Parameter address lower byte. Same as SPI Write data.
*2: Paramwter value
*3: Check sum (lower byte of "ADR + DAT)
*4: ST CODE(AA:in execution 55:EXEC.NORMALLY A5:EXEC.ERROR)

(2) CAMERA PARAMETER READ2(parameter address: 0x500 - 0x5FF)

SPI Write "data set"

Symbol	C1	C2	ADR	DAT	CS	ST
Value	00	01	*1	00	*2	AA

*1: Parameter address lower byte. Assignable value range is 00 to FF.
*2: Check sum (lower byte of "C1 + C2 + ADR + DAT")
*In READ command, DAT value is always set 00.

SPI Read "data set"(camera MCU response)

Symbol	ADR	DAT	CS	ST
Value	*1	*2	*3	*4

*1: Parameter address lower byte. Same as SPI Write data.
*2: Paramwter value
*3: Check sum (lower byte of "ADR + DAT")
*4: ST CODE(AA:in execution 55:EXEC.NORMALLY A5:EXEC.ERROR)

(3) CAMERA PARAMETER WRITE1(address: 0x400 - 0x4FF)

SPI Write "data set"

Symbol	C1	C2	ADR	DAT	CS	ST
Value	00	80	*1	*2	*3	AA

*1: Parameter address lower byte. Assignable value range is 00 to FF.

*3: Parameter Value.

*2: Check sum (lower byte of "C1 + C2 + ADR + DAT")

SPI Read "data set"(camera MCU response)

Symbol	ADR	DAT	CS	ST
Value	*1	*2	*3	*4

*1: Parameter address lower byte.(Same as SPI Write data.)

*2: Parameter value.(Same as SPI Write data.)

*3: Check sum (lower byte of "ADR + DAT")

*4: ST CODE(AA:in execution 55:EXEC.NORMALLY A5:EXEC.ERROR)

(4) CAMERA PARAMETER WRITE2(address: 0x500 - 0x5FF)

SPI Write "data set"

Symbol	C1	C2	ADR	DAT	CS	ST
Value	00	81	*1	*2	*3	AA

*1: Parameter address lower byte. Assignable value range is 00 to FF.

*2: Check sum (lower byte of "C1 + C2 + ADR + DAT)

*In READ command, DAT value is always set 00.

SPI Read "data set"(camera MCU response)

Symbol	ADR	DAT	CS	ST
Value	*1	*2	*3	*4

*1: Parameter address lower byte.(Same as SPI Write data.)

*2: Parameter value.(Same as SPI Write data.)

*3: Check sum (lower byte of "ADR + DAT")

*4: ST CODE(AA:in execution 55:EXEC.NORMALLY A5:EXEC.ERROR)

(5) AREA DISPLAY ON/OFF

SPI Write "data set"

Symbol	C1	C2	ADR	DAT	CS	ST
Value	01	*1	00	*2	*3	AA

*1: WINDOW SELECT(BLC = 0x00, HSBLC = 0x01)

*2: AREA DISPLAY ON: 0x01 / AREA DISPLAY OFF: 0x00

*3: Check sum (lower byte of "C1 + C2 + ADR + DAT")

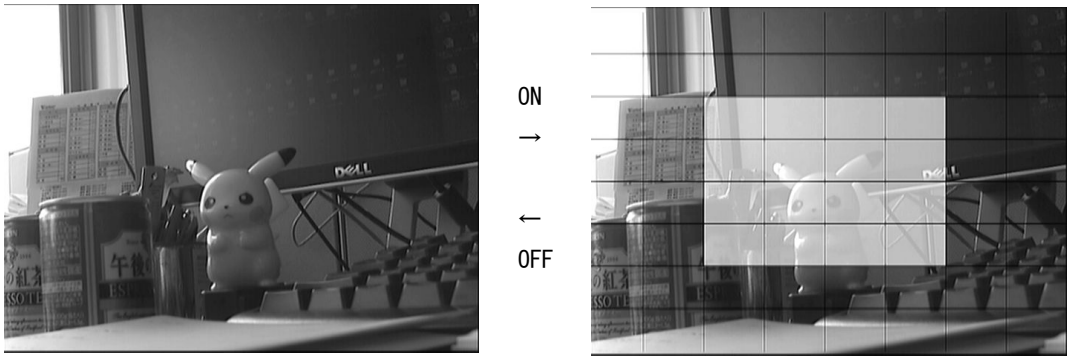
SPI Read "data set"(camera MCU response)

Symbol	ADR	DAT	CS	ST
Value	00	*4	*5	*6

*4: Parameter value.(Same as SPI Write data.)

*5: Check sum (lower byte of "ADR + DAT")

*6: ST CODE(AA:in execution 55:EXEC.NORMALLY A5:EXEC.ERROR)



(6) CAMERA OSD CONTROL(5 KEY OPERATION)

SPI Write "data set"

Symbol	C1	C2	ADR	DAT	CS	ST
Value	02	00	00	*1	*2	AA

*1: 5 key direction(See following table)

*2: Check sum (lower byte of "C1 + C2 + ADR + DAT)

5 key direction	DAT	CS
LEFT	01	03
RIGHT	02	04
UP	03	05
DOWN	04	06
SET	05	07

SPI Read "data set"(camera MCU response)

Symbol	ADR	DAT	CS	ST
Value	00	*3	*4	*5

*3: Parameter value.(Same as SPI Write data.)

*4: Check sum (lower byte of "ADR + DAT")

*5: ST CODE(AA:in execution 55:EXEC.NORMALLY A5:EXEC.ERROR)

(6) CAMERA PARAMETER SAVE

SPI Write "data set"

Symbol	C1	C2	ADR	DAT	CS	ST
Value	01	FF	00	00	00	AA

SPI Read "data set"(camera MCU response)

Symbol	ADR	DAT	CS	ST
Value	00	00	00	*1

*1: ST CODE(AA:in execution 55:EXEC.NORMALLY A5:EXEC.ERROR)

5.2.4 CAMERA PARAMETER

Please do not change the bit or the byte data with the notation of "-" among following tables.

(Camera operation will not be guaranteed.)

Address	Bit							
	7	6	5	4	3	2	1	0
0x400	-	-	-	-	-	-	-	-
0x401	-	-	-	-	AGC MODE 0x00:OFF 0x01:LOW 0x02:MID 0x03:HI		EI 0:OFF 1:ON	-
0x402	-	-	-	FIX SHUTTER SPEED 0x00:x256, 0x01:x128, 0x02:x64, 0x03:x32, 0x04:x16, 0x05:x8, 0x06:x4, 0x07:x2, 0x08:EI, 0x09:OFF(E:1/60,C:1/50), 0x0A:FL, 0x0B:1/250, 0x0C:1/500, 0x0D:1/1000, 0x0E:1/2000, 0x0F:1/5000, 0x10:1/10000, 0x11:1/100000				
0x403	-	-	-	-	-	-	-	-
0x404	-	-	-	-	-	-	-	-
0x405	-	-	-	-	-	-	-	-
0x406	-	-	-	-	SENS UP FUNCTION 0:OFF 1:ON	SENS UP MAX 0x00:x2, 0x01:x4, 0x02:x8, 0x03:x16, 0x04:x32, 0x05:x64, 0x06:x128, 0x07:x256(FLD)		
0x407	-	-	-	-	-	-	-	-
0x408	-	-	-	-	-	-	-	-
0x409	-	-	-	-	-	-	-	-
0x40A	-	-	-	-	-	-	-	-
0x40B	-	-	-	-	-	-	-	-
0x40C	-	-	-	-	-	-	-	-
0x40D	-	-	-	-	-	-	-	-
0x40E	-	-	-	-	-	-	-	-
0x40F	-	-	-	-	-	-	-	-
0x410	-	-	-	-	-	-	-	-
0x411	AGC LOW MAX actual AGC LOW MAX(dB) = ([0x05 … 0xFF] * 4 * 0.035) + 5.3 (notice 1) Be sure to set 5 or more large values. (notice 2) Be sure to set large value than AGC MIN.							
0x412	AGC MID MAX actual AGC MID MAX(dB) = ([0x05 … 0xFF] * 4 * 0.035) + 5.3 (notice 1) Be sure to set 5 or more large values. (notice 2) Be sure to set large value than AGC MIN.							
0x413	AGC HI MAX actual AGC HI MAX(dB) = ([0x05 … 0xFF] * 4 * 0.035) + 5.3 (notice 1) Be sure to set 5 or more large values. (notice 2) Be sure to set large value than AGC MIN.							
0x414	AGC MIN actual AGC MIN(dB) = ([0x05 … 0xFF] * 4 * 0.035) + 5.3 (notice 1) Be sure to set 5 or more large values. (notice 2) Be sure to set small value less than all AGC LO/MID/HI MAX.							
0x415	-	-	-	-	-	-	-	-
0x416	-	-	-	-	-	-	-	-

Address	Bit							
	7	6	5	4	3	2	1	0
0x417	—	—	—	—	MANUAL GAIN(AGC OFF GAIN) 0x01:6(dB) … 0x24:41(dB)			
0x418	—	—	—	—	HSBLC LEVEL 0x01:1(min) … 0x08:8(max)			
0x419	HSBLC WINDOW V START POSITION 0x00:0(TOP) … 0x07:7(BOTTOM)				HSBLC WINDOW H START POSITION 0x00:0(LEFT) … 0x07:7(RIGHT)			
0x420	HSBLC WINDOW V SIZE 0x01:1(min) … 0x08:8(max)				HSBLC WINDOW H SIZE 0x01:1(min) … 0x08:8(max)			
0x41B	—	—	—	—	BLC GAIN 0x00:LOW, 0x02:MID, 0x03:HI		BLC MODE 0x00:OFF, 0x01:BLC, 0x10:HSBLC	
0x41C	BLC WINDOW V START POSITION 0x00:0(TOP) … 0x07:7(BOTTOM)				BLC WINDOW H START POSITION 0x00:0(LEFT) … 0x07:7(RIGHT)			
0x41D	BLC WINDOW V SIZE 0x01:1(min) … 0x08:8(max)				BLC WINDOW H SIZE 0x01:1(min) … 0x08:8(max)			
0x41E	—	—	—	—	—	—	—	—
0x41F	—	—	—	—	—	—	—	—
0x420	—	—	—	—	—	—	—	—
0x421	—	—	—	—	—	—	—	—
0x422	—	—	—	—	—	—	—	—
0x423	—	—	—	—	—	—	—	—
0x424	—	—	—	—	—	—	—	—
0x425	—	—	—	—	—	—	—	—
0x426	—	—	—	—	—	—	—	—
0x427	—	—	—	—	—	—	—	—
0x428	—	—	—	—	—	—	—	—
0x429	—	—	—	—	—	—	—	—
0x42A	—	—	—	—	—	—	—	—
0x42B	—	—	—	—	—	—	—	—
0x42C	—	—	—	—	—	—	—	—
0x42D	—	—	—	—	—	—	—	—
0x42E	—	—	—	—	—	—	—	—
0x42F	—	—	—	—	—	—	—	—
0x430	—	—	—	—	—	—	—	—
0x431	—	—	—	—	—	—	—	—
0x432	—	—	—	—	—	—	—	—
0x433	—	—	—	—	—	—	—	—
0x434	—	—	—	—	—	—	—	—
0x435	—	—	—	—	—	—	—	—
0x436	—	—	—	—	—	—	—	—
0x437	—	—	—	—	—	—	—	—
0x438	—	—	—	—	—	—	—	—
0x439	—	—	—	—	—	—	—	—
0x43A	—	—	—	—	—	—	—	—
0x43B	—	—	—	—	—	—	—	—

Address	Bit							
	7	6	5	4	3	2	1	0
0x43C	—	—	—	—	—	—	—	—
0x43D	—	—	—	—	—	—	—	—
0x43E	—	—	—	—	—	—	—	—
0x43F	—	—	—	—	—	—	—	—
0x440	—	—	—	—	—	—	—	—
0x441	—	—	—	—	—	—	—	—
0x442	—	—	—	—	—	—	—	—
0x443	—	—	—	—	—	—	—	—
0x444	—	—	—	—	—	—	—	—
0x445	—	—	—	—	—	—	—	—
0x446	—	—	—	—	—	—	—	—
0x447	—	—	—	—	—	—	—	—
0x448	—	—	—	—	—	—	WDR MODE 0x00:OFF 0x01:USER1 0x02:USER2	
0x449	WDR USER1 H-LEVEL 0x00:0(LEVEL) … 0x0F:15(LEVEL)				WDR USER1 L-LEVEL 0x00:0(LEVEL) … 0x0F:15(LEVEL)			
0x44A	WDR USER2 H-LEVEL 0x00:0(LEVEL) … 0x0F:15(LEVEL)				WDR USER2 L-LEVEL 0x00:0(LEVEL) … 0x0F:15(LEVEL)			
0x44B	—	—	—	—	—	—	—	—
0x44C	—	—	—	—	—	—	—	—
0x44D	—	—	—	—	—	—	—	—
0x44E	—	—	—	—	—	—	—	—
0x44F	—	—	—	—	—	—	—	—
0x450	—	—	—	—	—	—	—	3DNR 0:OFF 1:ON
0x451	—	3DNR LEVEL 0x00:0(LEVEL) … 0x64:100(LEVEL)						
0x452	—	—	—	—	—	—	—	—
0x453	—	—	—	—	—	—	—	—
0x454	—	—	—	—	—	—	—	—
0x455	—	—	—	—	—	—	—	—
0x456	—	—	—	—	—	—	—	—
0x457	—	—	—	—	—	—	—	—
0x458	—	—	—	SHARPNESS 0x00:0(LEVEL) … 0x1F:31(LEVEL)				
0x459	—	—	—	—	—	—	—	—
0x45A	—	—	—	—	—	—	—	—
0x45B	—	—	—	—	—	—	—	—
0x45C	—	—	—	—	—	—	—	—
0x45D	—	—	—	—	—	—	—	—
0x45E	—	—	—	—	—	—	—	—
0x45F	—	—	—	—	—	—	—	—

Address	Bit							
	7	6	5	4	3	2	1	0
0x460	—	—	—	—	—	—	—	MOTION 0:OFF 1:ON
0x461	—	MOTION VIEW 0:OFF 1:ON	—	—	—	—	—	—
0x462	MOTION AREA1 SENSITIVITY 0x32:0, 0x31:1 … 0x28:10 … 0x1E:20 … 0x14:30 … 0x0B:39, 0x0A:40							
0x463	MOTION AREA2 SENSITIVITY 0x32:0, 0x31:1 … 0x28:10 … 0x1E:20 … 0x14:30 … 0x0B:39, 0x0A:40							
0x464	MOTION AREA3 SENSITIVITY 0x32:0, 0x31:1 … 0x28:10 … 0x1E:20 … 0x14:30 … 0x0B:39, 0x0A:40							
0x465	MOTION AREA4 SENSITIVITY 0x32:0, 0x31:1 … 0x28:10 … 0x1E:20 … 0x14:30 … 0x0B:39, 0x0A:40							
0x466	MOTION AREA1 START H POSITION 0x04:(LEFT) … 0xBB:(RIGHT) (note) Sum of "H POSITION" and "H size" is needed to be under 0xBF.							
0x467	MOTION AREA1 START V POSITION 0x01:(TOP) … 0x8F:(BOTTOM) (note) Sum of "V POSITION" and "V size" is needed to be under 0x90.							
0x468	MOTION AREA1 H SIZE 0x04:(min) … 0xBB:(max) (note) Sum of "H POSITION" and "H size" is needed to be under 0xBF.							
0x469	MOTION AREA1 V SIZE 0x01:(min) … 0x90:(max) (note) Sum of "V POSITION" and "V size" is needed to be under 0x90.							
0x46A	MOTION AREA2 START H POSITION 0x04:(LEFT) … 0xBB:(RIGHT) (note) Sum of "H POSITION" and "H size" is needed to be under 0xBF.							
0x46B	MOTION AREA2 START V POSITION 0x01:(TOP) … 0x8F:(BOTTOM) (note) Sum of "V POSITION" and "V size" is needed to be under 0x90.							
0x46C	MOTION AREA2 H SIZE 0x04:(min) … 0xBB:(max) (note) Sum of "H POSITION" and "H size" is needed to be under 0xBF.							
0x46D	MOTION AREA2 V SIZE 0x01:(min) … 0x90:(max) (note) Sum of "V POSITION" and "V size" is needed to be under 0x90.							
0x46E	MOTION AREA3 START H POSITION 0x04:(LEFT) … 0xBB:(RIGHT) (note) Sum of "H POSITION" and "H size" is needed to be under 0xBF.							
0x46F	MOTION AREA3 START V POSITION 0x01:(TOP) … 0x8F:(BOTTOM) (note) Sum of "V POSITION" and "V size" is needed to be under 0x90.							

Address	Bit							
	7	6	5	4	3	2	1	0
0x470	MOTION AREA3 H SIZE 0x04:(min) … 0xBB:(max) (note) Sum of "H POSITION" and "H size" is needed to be under 0xBF.							
0x471	MOTION AREA3 V SIZE 0x01:(min) … 0x90:(max) (note) Sum of "V POSITION" and "V size" is needed to be under 0x90.							
0x472	MOTION AREA4 START H POSITION 0x04:(LEFT) … 0xBB:(RIGHT) (note) Sum of "H POSITION" and "H size" is needed to be under 0xBF.							
0x473	MOTION AREA4 START V POSITION 0x01:(TOP) … 0x8F:(BOTTOM) (note) Sum of "V POSITION" and "V size" is needed to be under 0x90.							
0x474	MOTION AREA4 H SIZE 0x04:(min) … 0xBB:(max) (note) Sum of "H POSITION" and "H size" is needed to be under 0xBF.							
0x475	MOTION AREA4 V SIZE 0x01:(min) … 0x90:(max) (note) Sum of "V POSITION" and "V size" is needed to be under 0x90.							
0x476	—	—	—	—	MOTION AREA4 DISPLAY 0:OFF 1:ON	MOTION AREA3 DISPLAY 0:OFF 1:ON	MOTION AREA2 DISPLAY 0:OFF 1:ON	MOTION AREA1 DISPLAY 0:OFF 1:ON
0x477	—	—	—	—	—	—	—	—
0x478	—	—	—	—	—	—	—	—
0x479	—	—	—	—	—	—	—	—
0x47A	—	—	—	—	—	—	—	—
0x47B	—	—	—	—	—	—	—	—
0x47C	—	—	—	—	—	—	—	—
0x47D	—	—	—	—	—	—	—	—
0x47E	—	—	—	—	—	—	—	—
0x47F	—	—	—	—	—	—	—	—
0x480	—	—	—	—	—	—	—	—
0x481	—	—	—	—	—	—	—	—
0x482	—	—	—	—	—	—	—	—
0x483	—	—	—	—	—	—	—	—
0x484	—	—	—	—	—	—	—	—
0x485	—	—	—	—	—	—	—	—
0x486	—	—	—	—	—	—	—	—
0x487	—	—	—	—	—	—	—	—
0x488	—	—	—	—	—	—	—	—
0x489	—	—	—	—	—	—	—	—
0x48A	—	—	—	—	—	—	—	—
0x48B	—	—	—	—	—	—	—	—
0x48C	—	—	—	—	—	—	—	—
0x48D	—	—	—	—	—	—	—	—
0x48E	—	—	—	—	—	—	—	—
0x48F	—	—	—	—	—	—	—	—
0x490	—	—	—	—	—	—	—	—
0x491	—	—	—	—	—	—	—	—
0x492	—	—	—	—	—	—	—	—

Address	Bit							
	7	6	5	4	3	2	1	0
0x493	—	—	—	—	—	—	—	—
0x494	—	—	—	—	—	—	—	—
0x495	—	—	—	—	—	—	—	—
0x496	—	—	—	—	—	—	—	—
0x497	—	—	—	—	—	—	—	—
0x498	—	—	—	—	—	—	—	—
0x499	—	—	—	—	—	—	—	—
0x49A	—	—	—	—	—	—	—	—
0x49B	—	—	—	—	—	—	—	—
0x49C	—	—	—	—	—	—	—	—
0x49D	—	—	—	—	—	—	—	—
0x49E	—	—	—	—	—	—	—	—
0x49F	—	—	—	—	—	—	—	—
0x4A0	—	—	—	—	—	—	—	—
0x4A1	—	—	—	—	—	—	—	—
0x4A2	—	—	—	—	—	—	—	—
0x4A3	—	—	—	—	—	—	—	—
0x4A4	—	—	—	—	—	—	—	—
0x4A5	—	—	—	—	—	—	—	—
0x4A6	—	—	—	—	—	—	—	—
0x4A7	—	—	—	—	—	—	—	—
0x4A8	—	—	—	—	—	—	—	ZOOM 0:OFF 1:ON
0x4A9	ZOOM(magnification) 0x00:x1.0, 0x01:x1.1 … 0x09:x1.9, 0x0A:x2.0, 0x0B:x2.1 … 0x13:x2.9, 0x14:x3.0, 0x15:x3.1 … 0x1D:x3.9, 0x1E:x4.0, 0x1F:x4.5, 0x20:x5.0, 0x21:x5.5, 0x22:x6.0, 0x23:x6.5, 0x24:x7.0, 0x25:x7.5, 0x26:x8.0, 0x27:x9, 0x28:x10, 0x29:x11, 0x2A:x12, 0x2B:x13, 0x2C:x14, 0x2D:x15, 0x2E:x16, 0x2F:x18, 0x30:x20, 0x31:x22, 0x32:x24, 0x33:x26, 0x34:x28, 0x35:x30, 0x36:x32							
0x4AA	ZOOM(PAN) 0x00:-100(LEFT), 0x01:-99 … 0x63:-1, 0x64:0(CENTER), 0x65:+1 … 0xC7:+99, 0xC8:+100(RIGHT)							
0x4AB	ZOOM(TILT) 0x00:-100(UP), 0x01:-99 … 0x63:-1, 0x64:0(CENTER), 0x65:+1 … 0xC7:+99, 0xC8:+100(DOWN)							
0x4AC	—	—	—	—	—	—	—	—
0x4AD	—	—	—	—	—	—	—	—
0x4AE	—	—	—	—	—	—	—	—
0x4AF	—	—	—	—	—	—	—	—
0x4B0	—	—	—	—	NEG.IMAGE 0:OFF 1:ON	FLIP 0x00:OFF 0x01:H 0x02:V 0x03:HV		FREEZE 0:OFF 1:ON
0x4B1	—	—	—	GAMMA 0x00:USER, 0x01:0.05 … 0x09:0.45 … 0x14:1.0				
0x4B2	—	—	—	—	—	—	—	—
0x4B3	—	—	—	—	—	—	—	—
0x4B4	—	—	—	—	—	—	—	—
0x4B5	—	—	—	—	—	—	—	—
0x4B6	—	—	—	—	—	—	—	—

Address	Bit							
	7	6	5	4	3	2	1	0
0x4B7	—	—	—	—	—	—	—	—
0x4B8	—	—	—	—	—	—	—	—
0x4B9	—	—	—	—	—	—	—	—
0x4BA	—	—	—	—	—	—	—	—
0x4BB	—	—	—	—	—	—	—	—
0x4BC	—	—	—	—	—	—	—	—
0x4BD	—	—	—	—	—	—	—	—
0x4BE	—	—	—	—	—	—	—	—
0x4BF	—	—	—	—	—	—	—	—
0x4C0	—	—	—	—	—	—	—	—
0x4C1	—	—	—	—	—	—	—	—
0x4C2	—	—	—	—	—	—	—	—
0x4C3	—	—	—	—	—	—	—	—
0x4C4	—	—	—	—	—	—	—	—
0x4C5	—	—	—	—	—	—	—	—
0x4C6	—	—	—	—	—	—	—	—
0x4C7	—	—	—	—	—	—	—	—
0x4C8	—	—	—	—	—	—	—	—
0x4C9	—	—	—	—	—	—	—	—
0x4CA	—	—	—	—	—	—	—	—
0x4CB	—	—	—	—	—	—	—	—
0x4CC	—	—	—	—	—	—	—	—
0x4CD	—	—	—	—	—	—	—	—
0x4CE	—	—	—	—	—	—	—	—
0x4CF	—	—	—	—	—	—	—	—
0x4D0	—	—	—	—	—	—	—	—
0x4D1	—	—	—	—	—	—	—	—
0x4D2	—	—	—	—	—	—	—	—
0x4D3	—	—	—	—	—	—	—	—
0x4D4	—	—	—	—	—	—	—	—
0x4D5	—	—	—	—	—	—	—	—
0x4D6	—	—	—	—	—	—	—	—
0x4D7	—	—	—	—	—	—	—	—
0x4D8	—	—	—	—	—	—	—	—
0x4D9	—	—	—	—	—	—	—	—
0x4DA	—	—	—	—	—	—	—	—
0x4DB	—	—	—	—	—	—	—	—
0x4DC	—	—	—	—	—	—	—	—
0x4DD	—	—	—	—	—	—	—	—
0x4DE	—	—	—	—	—	—	—	—
0x4DF	—	—	—	—	—	—	—	—
0x4E0	—	—	—	—	—	—	—	—
0x4E1	—	—	—	—	—	—	—	—
0x4E2	—	—	—	—	—	—	—	—
0x4E3	—	—	—	—	—	—	—	—
0x4E4	—	—	—	—	—	—	—	—
0x4E5	—	—	—	—	—	—	—	—
0x4E6	—	—	—	—	—	—	—	—
0x4E7	—	—	—	—	—	—	—	—
0x4E8	—	—	—	—	—	—	—	—
0x4E9	—	—	—	—	—	—	—	—

Address	Bit							
	7	6	5	4	3	2	1	0
0x4EA	—	—	—	—	—	—	—	—
0x4EB	—	—	—	—	—	—	—	—
0x4EC	—	—	—	—	—	—	—	—
0x4ED	—	—	—	—	—	—	—	—
0x4EE	—	—	—	—	—	—	—	—
0x4EF	—	—	—	—	—	—	—	—
0x4F0	—	—	—	—	—	—	—	—
0x4F1	—	—	—	—	—	—	—	—
0x4F2	—	—	—	—	—	—	—	—
0x4F3	—	SETUP ON/OFF 0:0(IRE) 1:7.5(IRE)	SETUP LEVEL 0x00 … 0x0x3F 0x1C:0(IRE)/0x31:7.5(IRE)					
0x4F4	—	—	—	—	—	—	—	—
0x4F5	—	—	—	—	—	—	—	—
0x4F6	—	—	—	—	—	—	—	—
0x4F7	—	—	—	—	—	—	—	—
0x4F8	—	—	—	—	—	—	—	—
0x4F9	—	—	—	—	—	—	—	—
0x4FA	—	—	—	—	—	—	—	—
0x4FB	—	—	—	—	—	—	—	—
0x4FC	—	—	—	—	—	—	—	—
0x4FD	—	—	—	—	—	—	—	—
0x4FE	—	—	BPC LEVEL 0x01:1 … 0x04:4			BPC FLD 0x00:x4, 0x01:x8, 0x02:x16, 0x03:x32, 0x04:x64		
0x4FF	—	—	—	—	—	—	—	—

Address	Bit							
	7	6	5	4	3	2	1	0
0x500	—	—	—	—	—	—	—	—
0x501	—	—	—	—	—	—	—	—
0x502	—	—	—	—	—	—	—	—
0x503	—	—	—	—	—	—	—	—
0x504	—	—	—	—	—	—	—	—
0x505	—	—	—	—	—	—	—	—
0x506	—	—	—	—	—	—	—	—
0x507	—	—	—	—	—	—	—	—
0x508	—	—	—	—	—	—	—	—
0x509	—	—	—	—	—	—	—	—
0x50A	—	—	—	—	—	—	—	—
0x50B	—	—	—	—	—	—	—	—
0x50C	—	—	—	—	—	—	—	—
0x50D	—	—	—	—	—	—	—	—
0x50E	—	—	—	—	—	—	—	—
0x50F	—	—	—	—	—	—	—	—
0x510	—	—	—	—	—	—	—	—
0x511	—	—	—	—	—	—	—	—
0x512	—	—	—	—	—	—	—	—
0x513	—	—	—	—	—	—	—	—
0x514	—	—	—	—	—	—	—	—
0x515	—	—	—	—	—	—	—	—
0x516	—	—	—	—	—	—	—	—
0x517	—	—	—	—	—	—	—	—
0x518	—	—	—	—	—	—	—	—
0x519	—	—	—	—	—	—	—	—
0x51A	—	—	—	—	—	—	—	—
0x51B	—	—	—	—	—	—	—	—
0x51C	—	—	—	—	—	—	—	—
0x51D	—	—	—	—	—	—	—	—
0x51E	—	—	—	—	—	—	—	—
0x51F	—	—	—	—	—	—	—	—
0x520	—	—	—	—	—	—	—	—
0x521	—	—	—	—	—	—	—	—
0x522	—	—	—	—	—	—	—	—
0x523	—	—	—	—	—	—	—	—
0x524	—	—	—	—	—	—	—	—
0x525	—	—	—	—	—	—	—	—
0x526	—	—	—	—	—	—	—	—
0x527	—	—	—	—	—	—	—	—
0x528	—	—	—	—	—	—	—	—
0x529	—	—	—	—	—	—	—	—
0x52A	—	—	—	—	—	—	—	—
0x52B	—	—	—	—	—	—	—	—
0x52C	—	—	—	—	—	—	—	—
0x52D	—	—	—	—	—	—	—	—
0x52E	—	—	—	—	—	—	—	—
0x52F	—	—	—	—	—	—	—	—
0x530	—	—	—	—	—	—	—	—
0x531	—	—	—	—	—	—	—	—
0x532	—	—	—	—	—	—	—	—

Address	Bit							
	7	6	5	4	3	2	1	0
0x533	—	—	—	—	—	—	—	—
0x534	—	—	—	—	—	—	—	—
0x535	—	—	—	—	—	—	—	—
0x536	—	—	—	—	—	—	—	—
0x537	—	—	—	—	—	—	—	—
0x538	—	—	—	—	—	—	—	—
0x539	—	—	—	—	—	—	—	—
0x53A	—	—	—	—	—	—	—	—
0x53B	—	—	—	—	—	—	—	—
0x53C	—	—	—	—	—	—	—	—
0x53D	—	—	—	—	—	—	—	—
0x53E	—	—	—	—	—	—	—	—
0x53F	—	—	—	—	—	—	—	—
0x540	—	—	—	—	—	—	—	—
0x541	—	—	—	—	—	—	—	—
0x542	—	—	—	—	—	—	—	—
0x543	—	—	—	—	—	—	—	—
0x544	—	—	—	—	—	—	—	—
0x545	—	—	—	—	—	—	—	—
0x546	—	—	—	—	—	—	—	—
0x547	—	—	—	—	—	—	—	—
0x548	—	—	—	—	—	—	—	—
0x549	—	—	—	—	—	—	—	—
0x54A	—	—	—	—	—	—	—	—
0x54B	—	—	—	—	—	—	—	—
0x54C	—	—	—	—	—	—	—	—
0x54D	—	—	—	—	—	—	—	—
0x54E	—	—	—	—	—	—	—	—
0x54F	—	—	—	—	—	—	—	—
0x550	—	—	—	—	—	—	—	—
0x551	—	—	—	—	—	—	—	—
0x552	—	—	—	—	—	—	—	—
0x553	—	—	—	—	—	—	—	—
0x554	—	—	—	—	—	—	—	—
0x555	—	—	—	—	—	—	—	—
0x556	—	—	—	—	—	—	—	—
0x557	—	—	—	—	—	—	—	—
0x558	—	—	—	—	—	—	—	—
0x559	—	—	—	—	—	—	—	—
0x55A	—	—	—	—	—	—	—	—
0x55B	—	—	—	—	—	—	—	—
0x55C	—	—	—	—	—	—	—	—
0x55D	—	—	—	—	—	—	—	—
0x55E	—	—	—	—	—	—	—	—
0x55F	—	—	—	—	—	—	—	—
0x560	—	—	—	—	—	—	—	—
0x561	—	—	—	—	—	—	—	—
0x562	—	—	—	—	—	—	—	—
0x563	—	—	—	—	—	—	—	—
0x564	—	—	—	—	—	—	—	—
0x565	—	—	—	—	—	—	—	—

Address	Bit							
	7	6	5	4	3	2	1	0
0x566	—	—	—	—	—	—	—	—
0x567	—	—	—	—	—	—	—	—
0x568	—	—	—	—	—	—	—	—
0x569	—	—	—	—	—	—	—	—
0x56A	—	—	—	—	—	—	—	—
0x56B	—	—	—	—	—	—	—	—
0x56C	—	—	—	—	—	—	—	—
0x56D	—	—	—	—	—	—	—	—
0x56E	—	—	—	—	—	—	—	—
0x56F	—	—	—	—	—	—	—	—
0x570	—	—	—	—	—	—	—	—
0x571	—	—	—	—	—	—	—	—
0x572	—	—	—	—	—	—	—	—
0x573	—	—	—	—	—	—	—	—
0x574	—	—	—	—	—	—	—	—
0x575	—	—	—	—	—	—	—	—
0x576	—	—	—	—	—	—	—	—
0x577	—	—	—	—	—	—	—	—
0x578	—	—	—	—	—	—	—	—
0x579	—	—	—	—	—	—	—	—
0x57A	—	—	—	—	—	—	—	—
0x57B	—	—	—	—	—	—	—	—
0x57C	—	—	—	—	—	—	—	—
0x57D	—	—	—	—	—	—	—	—
0x57E	—	—	—	—	—	—	—	—
0x57F	—	—	—	—	—	—	—	—
0x580	—	—	—	—	—	—	—	—
0x581	—	—	—	—	—	—	—	—
0x582	—	—	—	—	—	—	—	—
0x583	—	—	—	—	—	—	—	—
0x584	—	—	—	—	—	—	—	—
0x585	—	—	—	—	—	—	—	—
0x586	—	—	—	—	—	—	—	—
0x587	—	—	—	—	—	—	—	—
0x588	—	—	—	—	—	—	—	—
0x589	—	—	—	—	—	—	—	—
0x58A	—	—	—	—	—	—	—	—
0x58B	—	—	—	—	—	—	—	—
0x58C	—	—	—	—	—	—	—	—
0x58D	—	—	—	—	—	—	—	—
0x58E	—	—	—	—	—	—	—	—
0x58F	—	—	—	—	—	—	—	—
0x590	—	—	—	—	—	—	—	—
0x591	—	—	—	—	—	—	—	—
0x592	—	—	—	—	—	—	—	—
0x593	—	—	—	—	—	—	—	—
0x594	—	—	—	—	—	—	—	—
0x595	—	—	—	—	—	—	—	—
0x596	—	—	—	—	—	—	—	—
0x597	—	—	—	—	—	—	—	—
0x598	—	—	—	—	—	—	—	—

Address	Bit							
	7	6	5	4	3	2	1	0
0x599	—	—	—	—	—	—	—	—
0x59A	—	—	—	—	—	—	—	—
0x59B	—	—	—	—	—	—	—	—
0x59C	—	—	—	—	—	—	—	—
0x59D	—	—	—	—	—	—	—	—
0x59E	—	—	—	—	—	—	—	—
0x59F	—	—	—	—	—	—	—	—
0x5A0	—	—	—	—	—	—	—	—
0x5A1	—	—	—	—	—	—	—	—
0x5A2	—	—	—	—	—	—	—	—
0x5A3	—	—	—	—	—	—	—	—
0x5A4	—	—	—	—	—	—	—	—
0x5A5	—	—	—	—	—	—	—	—
0x5A6	—	—	—	—	—	—	—	—
0x5A7	—	—	—	—	—	—	—	—
0x5A8	—	—	—	—	—	—	—	—
0x5A9	—	—	—	—	—	—	—	—
0x5AA	—	—	—	—	—	—	—	—
0x5AB	—	—	—	—	—	—	—	—
0x5AC	—	—	—	—	—	—	—	—
0x5AD	—	—	—	—	—	—	—	—
0x5AE	—	—	—	—	—	—	—	—
0x5AF	—	—	—	—	—	—	—	—
0x5B0	—	—	—	—	—	—	—	—
0x5B1	—	—	—	—	—	—	—	—
0x5B2	—	—	—	—	—	—	—	—
0x5B3	—	—	—	—	—	—	—	—
0x5B4	—	—	—	—	—	—	—	—
0x5B5	—	—	—	—	—	—	—	—
0x5B6	—	—	—	—	—	—	—	—
0x5B7	—	—	—	—	—	—	—	—
0x5B8	—	—	—	—	—	—	—	—
0x5B9	—	—	—	—	—	—	—	—
0x5BA	—	—	—	—	—	—	—	—
0x5BB	—	—	—	—	—	—	—	—
0x5BC	—	—	—	—	—	—	—	—
0x5BD	—	—	—	—	—	—	—	—
0x5BE	—	—	—	—	—	—	—	—
0x5BF	—	—	—	—	—	—	—	—
0x5C0	—	—	—	—	—	—	—	—
0x5C1	—	—	—	—	—	—	—	—
0x5C2	—	—	—	—	—	—	—	—
0x5C3	—	—	—	—	—	—	—	—
0x5C4	—	—	—	—	—	—	—	—
0x5C5	—	—	—	—	—	—	—	—
0x5C6	—	—	—	—	—	—	—	—
0x5C7	—	—	—	—	—	—	—	—
0x5C8	—	—	—	—	—	—	—	—
0x5C9	—	—	—	—	—	—	—	—
0x5CA	—	—	—	—	—	—	—	—
0x5CB	—	—	—	—	—	—	—	—

Address	Bit							
	7	6	5	4	3	2	1	0
0x5CC	—	—	—	—	—	—	—	—
0x5CD	—	—	—	—	—	—	—	—
0x5CE	—	—	—	—	—	—	—	—
0x5CF	—	—	—	—	—	—	—	—
0x5D0	—	—	—	—	—	—	—	—
0x5D1	—	—	—	—	—	—	—	—
0x5D2	—	—	—	—	—	—	—	—
0x5D3	—	—	—	—	—	—	—	—
0x5D4	—	—	—	—	—	—	—	—
0x5D5	—	—	—	—	—	—	—	—
0x5D6	—	—	—	—	—	—	—	—
0x5D7	—	—	—	—	—	—	—	—
0x5D8	—	—	—	—	—	—	—	—
0x5D9	—	—	—	—	—	—	—	—
0x5DA	—	—	—	—	—	—	—	—
0x5DB	—	—	—	—	—	—	—	—
0x5DC	—	—	—	—	—	—	—	—
0x5DD	—	—	—	—	—	—	—	—
0x5DE	—	—	—	—	—	—	—	—
0x5DF	—	—	—	—	—	—	—	—
0x5E0	—	—	—	—	—	—	—	BPC 0:(OFF) 1:START
0x5E1	—	—	—	—	—	—	—	FACTORY RESET 0:OFF 1:ON
0x5E2	—	—	—	—	—	—	—	DIGIT OUT 0:OFF 1:ON
0x5E3	—	—	—	—	—	—	—	—
0x5E4	—	—	—	—	—	—	—	—
0x5E5	—	—	—	—	—	—	—	—
0x5E6	—	—	—	—	—	—	—	—
0x5E7	—	—	—	—	—	—	—	—
0x5E8	—	—	—	—	—	—	—	—
0x5E9	—	—	—	—	—	—	—	—
0x5EA	—	—	—	—	—	—	—	—
0x5EB	—	—	—	—	—	—	—	—
0x5EC	—	—	—	—	—	—	—	—
0x5ED	—	—	—	—	—	—	—	—
0x5EE	—	—	—	—	—	—	—	—
0x5EF	—	—	—	—	—	—	—	—
0x5F0	—	—	—	—	—	—	—	—
0x5F1	—	—	—	—	—	—	—	—
0x5F2	—	—	—	—	—	—	—	—
0x5F3	—	—	—	—	—	—	—	—
0x5F4	—	—	—	—	—	—	—	—
0x5F5	—	—	—	—	—	—	—	—
0x5F6	—	—	—	—	—	—	—	—
0x5F7	—	—	—	—	—	—	—	—
0x5F8	—	—	—	—	—	—	—	—

Address	Bit							
	7	6	5	4	3	2	1	0
0x5F9	—	—	—	—	—	—	—	—
0x5FA	—	—	—	—	—	—	—	—
0x5FB	—	—	—	—	—	—	—	—
0x5FC	—	—	—	—	—	—	—	—
0x5FD	—	—	—	—	—	—	—	—
0x5FE	—	—	—	—	—	—	—	—
0x5FF	—	—	—	—	—	—	—	—

5.2.4 SAMPLE CODE

The following pages are example of the camera parameter change by SPI communication.
In the examples, AGC mode is changed by using the method of 3WIRE SPI communication.

(1)SAMPLE CODE

This sample source code is installed in and tested on "R8C/1B" MCU made by Renesas.
Therefore, it is necessary to change source code into the specification of the microcomputer to be used.

```
//-----------------------------------------------------------------------
//
//  FILE        : SPI_TEST_3W.c
//                SPI communication test program
//  DATE        : 2012.02.27
//  DESCRIPTION : Test program to change camera parameters
//                by using MCU GPIO.
//                (SPI 3wire communication test)
//  Watec  S.Igarashi
//  CPU GROUP   :1B
//
//-----------------------------------------------------------------------

#include    sfr_r81b.h

#define     TRUE       1
#define     FALSE      0
#define     CLKMS    500            // for wait counter

//  global variables-------------------------------------------------------
unsigned char SEND_BUFF[0x10] ;      // Send Buffer(16 byte)
unsigned char READ_BUFF[0x10] ;      // Receive Buffer(16 byte)
unsigned char SW_STATUS ;            // SW input value
unsigned char SW_S_C[4] ;            // temporary variables for avoid SW chattering
unsigned int  Adr_C  = 0x400 ;       // Address counter

// prototyping ----------------------------------------------------------
// checking to which SW is ON
unsigned char    check_key_status(void) ;

// SPI Write (3wire)
unsigned char    SPI_WRITE_3W(unsigned char, unsigned char) ;

// SPI Read (3wire)
unsigned char    SPI_READ_3W(unsigned char, unsigned char) ;

// CAMERA CONTROL SUB ROUTINES
unsigned char    AGC_MODE(void) ;
unsigned char    BLC_AREA_ON(void) ;
unsigned char    BLC_AREA_OFF(void) ;
unsigned char    INDIRECT_READ(unsigned int) ;
// etc.
void             wait_ms(unsigned int) ;
void             wait_nop(void) ;
```

```
// ---- main loop ------------------------------------------------------
void main(void)
{
    int     i ;                     // loop counter

    // initialize MCU
    // inhibit interrupt(INT0)
    int0en  = 0 ;
    prc0    = 1 ;
    hra00   = 1 ;
    cm06    = 0 ;
    wait_nop() ;
    hra01   = 1 ;
    ocd2 = 1 ;

    // initialize GPIO port -------------------------------------------
    // port direction (input = 0 / output = 1)
    // port in group1(P1) are all SW input
    pd1_0 = 0 ;             // UP
    pd1_1 = 0 ;             // DOWN
    pd1_2 = 0 ;             // LEFT
    pd1_3 = 0 ;             // RIGHT
    pd1_4 = 0 ;             // SET
    pd1_5 = 0 ;             // DIP1
    pd1_6 = 0 ;             // DIP2
    pd1_7 = 0 ;             // DIP3

    // port I/O setting
    // P3_3,4,5,7 are using SPI communication
    pd3_3 = 0 ;             // MISO/SIMO(3wire)
    pd3_4 = 1 ;             // SCLK
    pd3_5 = 1 ;             // SS
    pd3_7 = 1 ;
    pd4_5 = 1 ;             // LED for status display

    // initialize output port
    p3_4 = 1 ;              // SCLK = H
    p3_5 = 1 ;              // SS = H
    p3_7 = 1 ;
    p4_5 = 1 ;              // LED OFF

    // initialize SW input port ALL OFF (ON = L)
    p1 = 0xFF ;

    //----- MAIN LOOP -------------------------------------------------------
    while(1){
        i = 0 ;                         // reset counter
        while(i<3){                     // read SW status in 3 times
            SW_S_C[i] = p1 ;
            wait_ms(10) ;               // 5ms waite
            SW_S_C[i+1] = p1 ;          // read one more
            if (SW_S_C[i] == SW_S_C[i+1]){
                i++ ;
            }else{
                i = 0 ;
            }
        }
```

```
        // if SW status equal 3 times, check previous SW status.
        // if it is not equal, key input was changed and stable.
        if(SW_STATUS != SW_S_C[3]){
            SW_STATUS = SW_S_C[3] ;
            // judge which SW is ON and processing SPI communication.
            if(check_key_status() != TRUE){
                p4_5 = 0 ;              // LED lit ON
            }else{
                p4_5 = 1 ;
            }
        }
    }
}

// end of main-------------------------------------------------------------------
//-------------------------------------------------------------------------------
// FUNCTION: check_key_status
//
//  Decode SW status and process SPI communication.
//  if illegal code was read, return FALSE.
//-------------------------------------------------------------------------------
unsicheck_key_status(void)
{
    unsigned    char    i ;
    unsigned    char    RetV ;          // TRUE/FALSE

    // decode 5 key
    switch( SW_STATUS & 0x1F ){
        case 0x1E:                      // 5key UP
            // INC address and Read Parameter
            if (Adr_C == 0x5FF){
                Adr_C = 0x400 ;
            }else{
                Adr_C++ ;
            }
            RetV = INDIRECT_READ(Adr_C) ;
            break ;
        case 0x1D:                      // 5key DOWN
            // DEC adress and Read Parameter
            if (Adr_C == 0x400){
                Adr_C = 0x5FF ;
            }else{
                Adr_C-- ;
            }
            RetV = INDIRECT_READ(Adr_C) ;
            break ;
        case 0x1B:                      // 5key LEFT
            RetV = BLC_AREA_OFF() ;
            break ;
        case 0x17:                      // 5key RIGHT
            RetV = BLC_AREA_ON() ;
            break ;
        case 0x0F:                      // 5key SET
            // AGC MODE change
            RetV = AGC_MODE() ;
            break ;
        default:
```

```
            RetV = TRUE ;               // nothing to do
            break ;
    }

    return RetV ;
}

unsigned char AGC_MODE(void)
{
    unsigned char       AGC_V ;         // AGC MAX = 0:OFF/1:LOW/2:MID/3:HI
    unsigned char       RetV ;          // temp

    // AGC MAX
    // ADDRESS 0x401 READ
    SEND_BUFF[2] = 0x00 ;
    SEND_BUFF[3] = 0x00 ;
    SEND_BUFF[4] = 0x01 ;
    SEND_BUFF[5] = 0x00 ;
    SEND_BUFF[6] = 0x01 ;
    SEND_BUFF[7] = 0xAA ;
    if (SPI_WRITE_3W(0x36, 8) == FALSE ){
        retu FALSE ;
    }else{
        RetV = SPI_READ_3W(0x38, 4) ;
    }
                                            // AGC MODE value (bit3-2)
    AGC_V = (READ_BUFF[1] & 0x0C) >> 2 ;    // mask 00001100 and shift
    if (AGC_V == 3){                        // 0->1->2->3->0
        AGC_V =0;
    }else{
        AGC_V++ ;
    }
    wait_ms(1) ;

    // ADDRESS 0x401 WRITE
    SEND_BUFF[2]// BANK-0
    SEND_BUFF[3] = 0x80 ;
    SEND_BUFF[4] = 0x01 ;
    SEND_BUFF[5] = (READ_BUFF[1] & 0xF3) + (AGC_V << 2) ;
    SEND_BUFF[6] = SEND_BUFF[2] + SEND_BUFF[3] + SEND_BUFF[4] + SEND_BUFF[5] ;
    SEND_BUFF[7] = 0xAA ;
    return SPI_WRITE_3W(0x36, 8) ;
}

unsigned char BLC_AREA_ON(void)
{

    SEND= 0x01 ;
    SEND= 0x00 ;
    SEND= 0x00 ;
    SEND= 0x01 ;                            // BLC AREA ON
    SEND= 0x02 ;
    SEND= 0xAA ;

    return SPI_WRITE_3W(0x36, 8) ;
}
```

```
unsigned char BLC_AREA_OFF(void)
{

    SEND= 0x01 ;
    SEND= 0x00 ;
    SEND= 0x00 ;
    SEND= 0x00 ;                                //BLC AREA OFF
    SEND= 0x01 ;
    SEND= 0xAA ;

    return SPI_WRITE_3W(0x36, 8) ;
}

unsigned char INDIRECT_READ(unsigned int Pm_Adr)
{
    unsigned char Bank ;
    unsigned char Adrs ;

    Bank = (unsigned char)(Pm_Adr >> 8) - 0x04 ;
    Adrs = (unsigned char)(Pm_Adr) ;

    SEND_BUFF[2]    = 0x00 ;
    SEND_BUFF[3]    = Bank ;
    SEND_BUFF[4]    = Adrs ;
    SEND_BUFF[5]    = 0x00 ;
    SEND_BUFF[6]    = (unsigned char)(Bank + Adrs) ;
    SEND_BUFF[7]    = 0xAA ;

    if (SPI_WRITE_3W(0x36, 8) == TRUE ){
        return SPI_READ_3W(0x38, 4) ;
    }else{
        return FALSE ;
    }
}

//---------------------------------------------------------------------------
// FUNCTION: SPI_WRITE_3W
//
//  SPI WRITE SEND_BUFF[0...(W_Byte-1)]
//  In case of Indirect access, it must W_Adr = 0x36 and W_Byte = 8.
//  In case of 3Wire, it need change GPIO(MISO/MOSI) direction
//  input/output.
//---------------------------------------------------------------------------
unsigned char   SPI_WRITE_3W(unsigned char W_Adr, unsigned char W_Byte)
{
    int i ;
    int j ;
    int WriteCMD ;
    unsigned char   SendBit ;

    // make SPI WRITE COMMAND
    WriteCMD = (int)W_Adr << 1 ;
    SEND_BUFF[0] = (unsigned char)WriteCMD ;            // lower 8bit
    SEND_BUFF[1] = (unsigned char)(WriteCMD >> 8) ;     // higher 8bit

    i = 0 ;             // Byte Counter
```

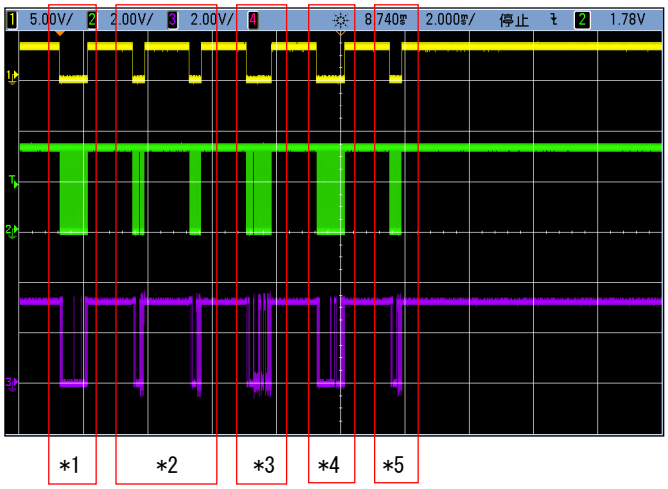
```
    // initialize SPI bus
    p3_5 = 1 ;          // SS   = H
    p3_4 = 1 ;          // SCLK = H

    wait_nop() ;

    // SPI WRITE COMMAND send start
    pd3_3 = 1;                                      // set port direction MOSI(output)
    p3_3 = 1 ;                                      // MOSI = H
    p3_5 = 0 ;                                      // SS   = L
    while( i < W_Byte ){                            // W_Byte must 8 in INDIRECT ACCESS
        j = 1 ;                                     // Bit Mask
        while(j < 0xFF){                            // 8bit send
            SendBit = SEND_BUFF[i] & j ? 1 : 0 ;    // bit test
            j = j << 1 ;                            // j *= 2 ;
            p3_4 = 0 ;                              // SCLK = L
            p3_3 = SendBit ;                        // output LSB first bit data on MOSI
            wait_nop() ;
            p3_4 = 1 ;                              // SCLK = H(latch)
        }
        i ++ ;                                      // Next byte
        wait_nop() ;
    }

    p3_5 = 1 ;                                      // SS = H(END)

    pd3_3 = 0;                                      // set port direction MISO(input)

    wait_ms(1) ;

    // CTL-COMMAND execution is done ?
    READ_BUFF[0] = 0xAA ;                           // initialize
    while( READ_BUFF[0] == 0xAA ){                  // 0xAA means busy
        SPI_READ_3W(0x3B, 1) ;                      // ST CODE read
        wait_ms(1) ;                                // 1mS wait
    }
    if( READ_BUFF[0] == 0x55){                      // 0x55 means EXEC.NORMALLY
        return TRUE ;
    }else{
        return FALSE ;
    }

}

//---------------------------------------------------------------------
// FUNCTION: SPI_READ_3W
//
//  SPI WRITE READ_BUFF[0...(R_Byte-1)]
//  In case of 3Wire, it need change GPIO(MISO/MOSI) direction
//  input/output.
//---------------------------------------------------------------------
unsigned char SPI_READ_3W(unsigned char R_Adr, unsigned char R_Byte)
{
    int     ReadCMD ;                               // SPI READ COMMAND
    int i ;                                         // counter
    int j ;
    unsigned char   SendBit ;
```

```
    // make SPI READ COMMAND
    ReadCMD = ((int)R_Adr << 1) + 1 ;
    SEND_BUFF[0] = (unsigned char)ReadCMD ;          // lower 8bit
    SEND_BUFF[1] = (unsigned char)(ReadCMD >> 8) ;   // higher 8bit

    i = 0 ;                                         // Byte Counter

    // initialize SPI bus
    p3_5 = 1 ;                                      // SS   = H
    p3_4 = 1 ;                                      // SCLK = H

    wait_nop() ;

    // SPI READ COMMAND send start
    pd3_3 = 1;                                      // set port direction MOSI(output)
    p3_3 = 1 ;                                      // MOSI = H
    p3_5 = 0 ;                                      // SS   = L
    wait_nop() ;
    while( i < 2 ){                                 // 2 byte loop
        j = 1 ;                                     // bit mask
        while(j < 0xFF){                            // 8bit send
            SendBit = SEND_BUFF[i] & j ? 1 : 0 ;    // Bit Test
            j = j << 1 ;                            // j *= 2 ;
            p3_4 = 0 ;                              // SCLK = L
            p3_3 = SendBit ;                        // output LSB first bit data on MOSI
            wait_nop() ;
            p3_4 = 1 ;                              // SCLK = H(latch)
        }
        i ++ ;                                      // next byte
        wait_nop() ;
    }

    // SPI READ start
    pd3_3 = 0 ;                                     // set port direction MISO(input)

    wait_nop() ;
    wait_nop() ;
    wait_nop() ;
    wait_nop() ;
    i = 0 ;
    while( i < R_Byte ){                            // Loop R_Byte times
        j = 0 ;                                     // reset bit counter
        READ_BUFF[i] = 0x00 ;                       // clear buffer
        while(j < 8){                               // 8bit receive
            p3_4 = 0 ;                              // SCLK = L(Data Change)
            wait_nop() ;
            wait_nop() ;
            p3_4 = 1 ;                              // SCLK = H(latch)
            // LSB first receive from MISO
            READ_BUFF[i] += (unsigned char)p3_3 << j ;
            j ++ ;                                  // Next Bit
        }
        i ++ ;                                      // Next Byte
    }
    p3_5 = 1 ;                                      // SS = H(END)
    return TRUE ;
}
```

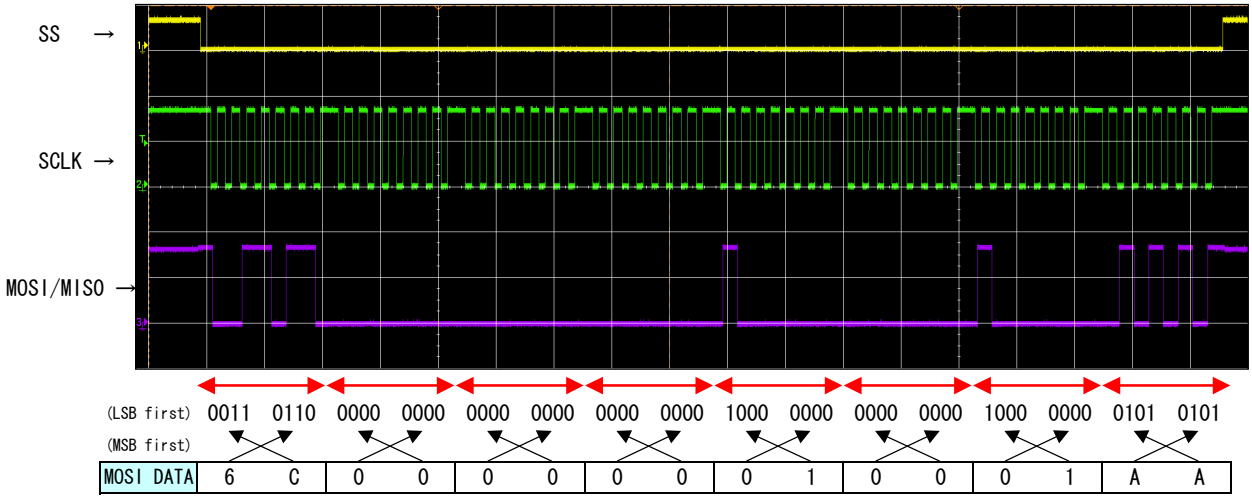
```
//---------------------------------------
// msec wait
//---------------------------------------
void    wait_ms(unsigned int wms)
{
    unsigned int w_cnt ;

    while ( wms--){
        w_cnt = CLKMS ;
        while(w_cnt--){}
    }
}

void    wait_nop(void)
{
    asm( "nop" ) ;
    asm( "nop" ) ;
    asm( "nop" ) ;
    asm( "nop" ) ;
    asm( "nop" ) ;
    asm( "nop" ) ;
}
```

(2)SAMPLE WAVEFORM

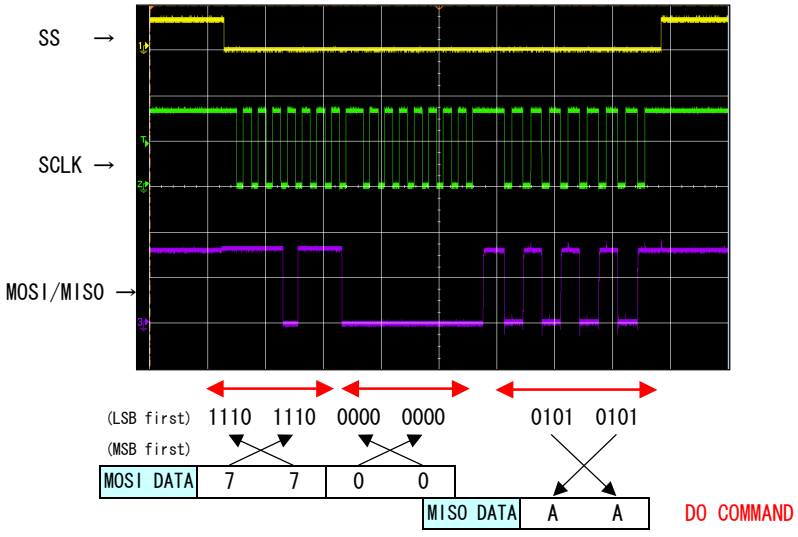
e.g. Change AGC MODE (MID -> HI) by using SPI(3wire) communucation.



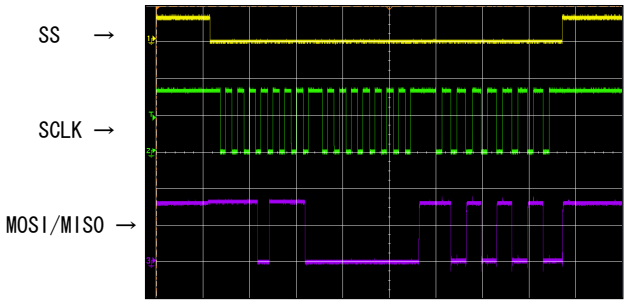
Step *1: Send "CAMERA PARAMETER READ1 command" to indirect registers(CAMERA: Address 0x36 - 0x3B) to get "AGC MODE: Address=0x401" parameter from CAMERA.
Following wave form is above operation by using "SPI write command".



Step *2: Send "SPI read command" to get and check ST CODE(address:0x3B).

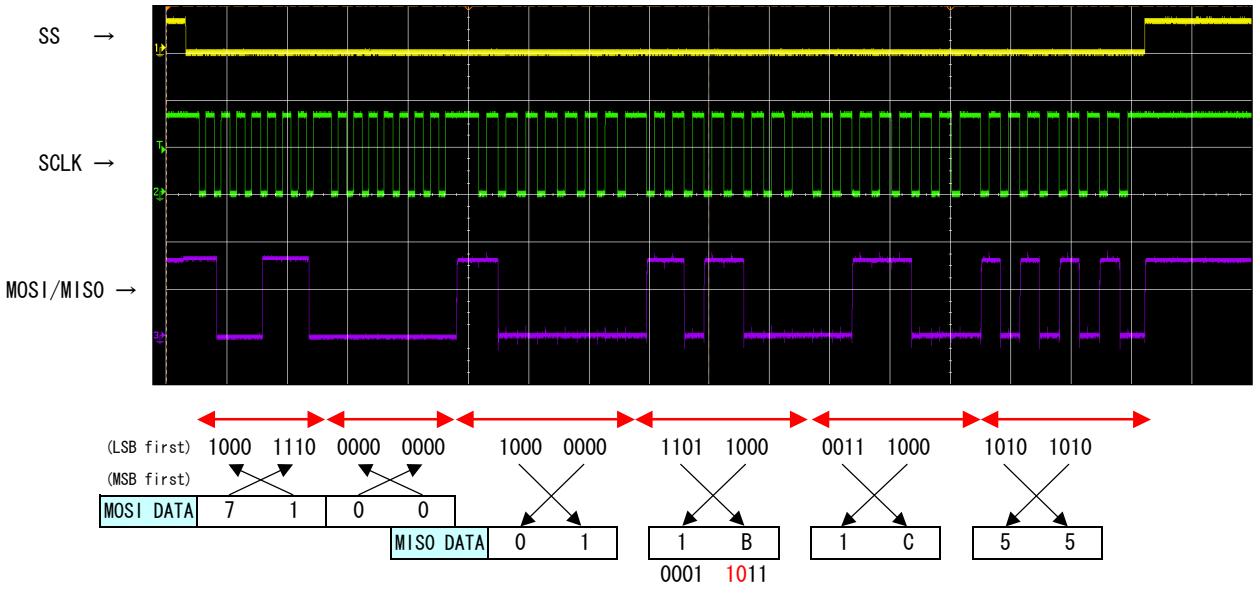


In this example, "ST CODE" is still "0xAA". It means that CAMERA MCU is busy.
Therefore, after about 1(mS) waiting, It need retrying.



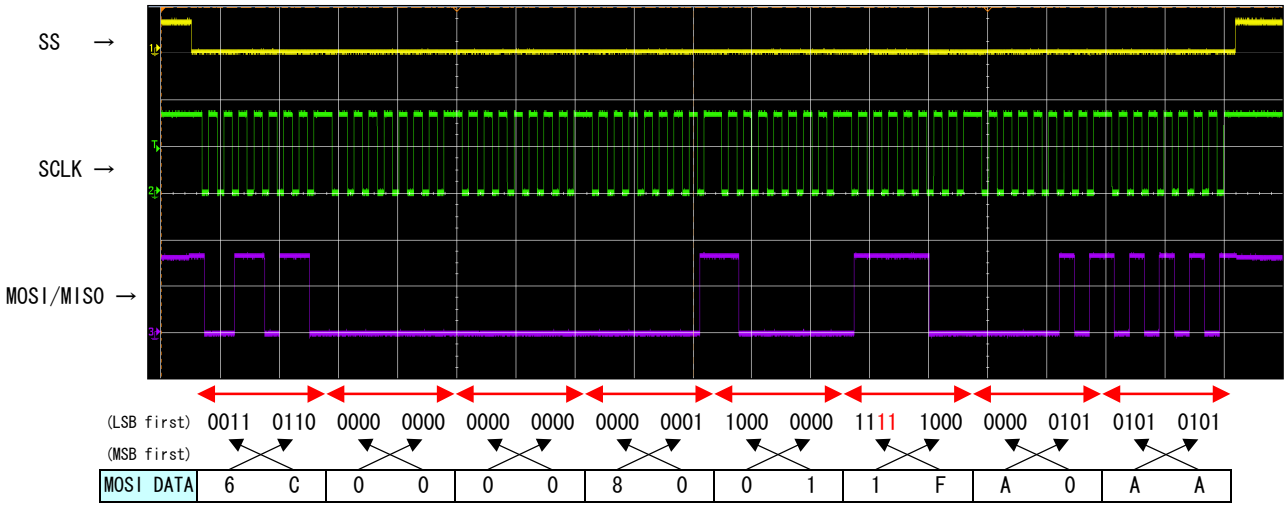
← Retry and get ST CODE = EXEC.NORMALLY.

Step *3: By using "SPI read command", get "AGC MODE: Address=0x401" parameter value from CAMERA.



"AGC MODE: Address=0x401, Bit 4-3" is 2. It means "AGC MODE = MID".
To change AGC MODE = HI, set 1 to Bit4 and 1 to Bit 3.
Therefore, prepare data (0x1F=00011111) and send to CAMERA.

Step *4: Send "CAMERA PARAMETER WRITE1 command" to indirect registers(CAMERA: Address 0x36 - 0x3B) to set "AGC MODE: Address=0x401" parameter to CAMERA.
Following wave form is above operation by using "SPI write command".



Step *5: Send "SPI read command" to get and check ST CODE(address:0x3B).