Noritake itron

Vacuum Fluorescent Display Module
“Program Macro”
Specification

Model: GU-3900B series

Specification No:	DS-1600-0006-01
Date of Issue:	October 22, 2010 (00)
Revision:	October 27, 2010 (01)

Published by
NORITAKE ITRON Corp. / Japan
http://www.noritake-itron.jp

Contents

1 General Description

This specification covers the GU-3900B series Program Macro feature.
Program Macro is a simple programming language for users to define arbitrary sequences of operation.

1.1 Program Macro Instructions

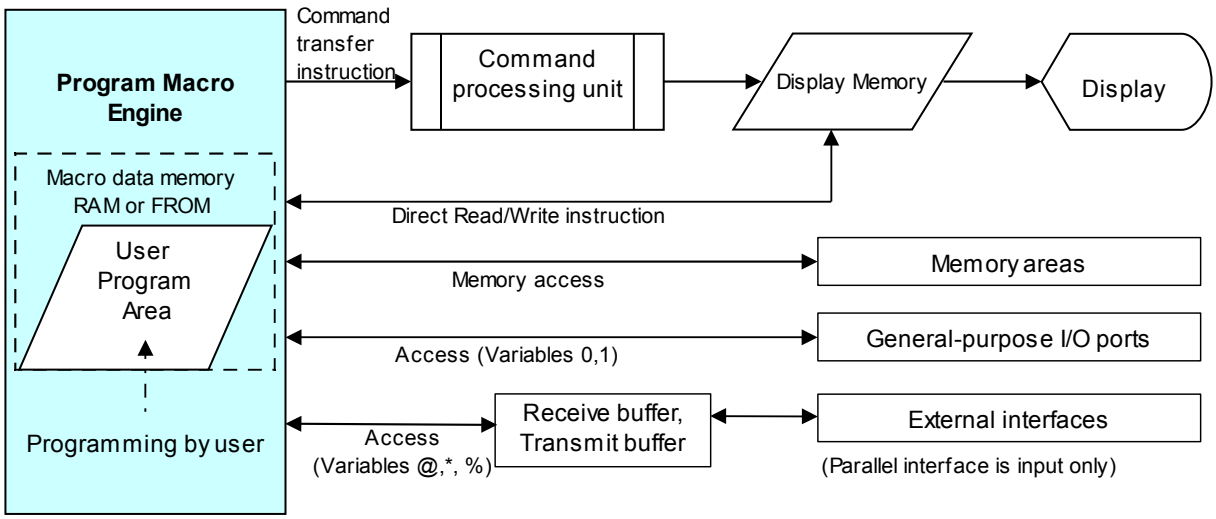
- Variable setting
- Arithmetic operations
- Conditional jump
- Unconditional jump
- Display Memory read/write
- Command transfer
- Subroutine call / return
- Wait
- Event handling
- etc

1.2 Control Functions

External interfaces, General-purpose I/O ports, and various memory areas can be controlled.

1.3 Program Macro operation overview

The Program Macro engine reads the program defined in the user program area, sends commands to the command processing unit, and accesses the General-purpose I/O ports, external interfaces, Display Memory and various other memory areas.



2 Program Macro Usage

2.1 Program Macro Definition

Program Macro is defined using either the RAM Macro define / delete command or the FROM Macro define / delete command.

2.2 Program Macro Start

Program macro is started using the Macro execution command.
Note: Program Macro is different to normal Macro in that execution is not continuously repeated.

Definition number and execution numbers correspond as follows:

Type	Macro define command, Macro definition number		Macro execution command, Macro definition number	
	RAM Macro	FROM Macro	Execute as normal Macro	Execute as Program Macro
RAM macro	(Only RAM Macro applicable)	-	00h	80h
FROM macro 1	-	01h	01h	81h
FROM macro 2	-	02h	02h	82h
FROM macro 3	-	03h	03h	83h
FROM macro 4	-	04h	04h	84h

2.3 Program Macro End

Program Macro is ended by the “Program Macro End” (BK) instruction.
Unlike normal Macro, Program Macro does not end if a command is input during execution.
Program Macro does not end until the Program Macro End instruction (BK) is executed.
Auto-start of Program Macro at power-on is available by setting Memory SW. If, due to a programming error, Program Macro End (BK) is never executed, it can become impossible to change the program. To recover from this situation, proceed as follows:

1. Power OFF
2. Set TEST terminal to 0 (low) and power ON to enter TEST mode.
3. Release TEST terminal to 1 (high) to end TEST mode.
4. Delete or re-program Program Macro.

Refer to the “Connectors” section in the Hardware specification for details on TEST terminal.

2.4 Program Macro Error

During execution of a Program Macro, if an inappropriate operation is encountered, a Program Macro error occurs. If a Program Macro error occurs, the screen is cleared and “Program Macro Error” is displayed in the top left of the screen. The Program Macro is stopped, and the module returns to Normal command mode. The error should then be fixed, then the Program Macro re-defined with the corrected code.

3 Programming Language

3.1 Variables

REG1 – REGn: Variable name (unsigned)

Variable name (code)	Contents	Read	Write	Initial value
A – Z (41h–5Ah)	Value of variable A – Z (16-bit / 32-bit)	○	○	Undefined
a – z (61h–7Ah)	Memory data at address pointed to by value of variable a–z. Memory area determined by Memory Access Area Setting.	○	○ (FROM ×)	Undefined
0,1 (30h, 31h)	General-purpose I/O port (only lower 8 bits used) (1)	○	○	Undefined
2 (32h)	DIP-SW (only lower 8 bits used)	○ (5)	-	DIP-SW setting
* (2Ah)	Receive buffer peek (only lower 8 bits used)	○ (4)	-	Empty
@ (40h)	Transmit / Receive buffer (only lower 8 bits used)	○ Rx buffer (2)	○ Tx buffer (3)	Empty
% (25h)	Receive buffer data length	○	×	0
? (3Fh)	Timer counter value (only lower 16 bits used)	○	○	0

(1): General-purpose I/O port input / output switching:
When switching from Input to Output, set the output value then change. Port direction is changed using the “I/O Port Input / Output setting” command (refer to “General Function” Software Specification).

(2): Reading the receive buffer blocks if there is no data in the receive buffer:
This can be avoided by first checking the receive buffer data length (Variable '%').

(3): Writing to the transmit buffer blocks if the transmit buffer is full. Transmit buffer capacity is 128 bytes.

(4): Only reads the data, and does not affect the receive buffer read counter.

(5): For each switch, '0' is read if ON, and '1' is read if OFF.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SW8	SW7	SW6	SW5	SW4	SW3	SW2	SW1

3.2 Data length

Data length for Variables and numeric values is set to either 16 bits or 32 bits using the “Variable data length set” instruction.
Note: Changing the data length during a program could cause unintended operation, so this should be set before using any variables.

3.3 Instruction set overview

Instruction name	Hex Code	Operation	Details	Example
Command transfer (Numeric)	50h,23h,nL,nH, d1,d2, ... dn	Transfer data d1...dn to Command processing unit. nL: Data sequence length, lower byte nH: Data sequence length, upper byte d1...dn: Data sequence	p8	p15
Command transfer (Variable)	50h,52h,nL,nH, REG1,REG2,... REGn	Transfer the contents of Variables REG1...REGn to Command processing unit. nL: Variable sequence length, lower byte nH: Variable sequence length, upper byte REG1...REGn: Variable sequence	p8	p15
Arithmetic operation (Variable← Variable, Numeric)	4Ch,23h,REG1, mode, d1,d2 [,d3,d4]	Perform calculation on REG1 with numerical value 'd' and store the result in REG1. REG1: Variable to store result mode: Operation mode = : Assign (3Dh) + : Add (2Bh) - : Subtract (2Dh) * : Multiply (2Ah) / : Divide (2Fh) %: Remainder (25h) | : Bit-wise OR (7Ch) & : Bit-wise AND (26h) ^ : Bit-wise XOR (5Eh) d1: Numeric value × 01h d2: Numeric value × 0100h d3: Numeric value × 010000h (if 32-bit) d4: Numeric value × 01000000h (if 32-bit)	p8	p16
Shift variable	4Ch,23h, REG1,mode,d1	Shift the contents of REG1 by d1 bits. REG1: Variable mode: Operation mode >: Right shift (3Eh) <: Left shift (3Ch) d1: Shift bit count	p9	p16
Increment / Decrement variable	4Ch,23h,REG1,mode	Increment or Decrement the contents of REG1. REG1: Variable mode: Operation mode i: Increment (69h) d: Decrement (64h)	p9	p16
Arithmetic operation (Variable1← Variable1, Variable2)	4Ch,52h, REG1,mode,REG2	Perform calculation on REG1 with REG2 and store the result in REG1. REG1: Variable to store result mode: Operation mode REG2: Operation variable 2	p9	p16
Push variable	50h,53h,REG1	Store the value in Variable REG1 onto the stack.	p9	p17
Pop variable	50h,4Ch,REG1	Retrieve a value off the stack into REG1.	p9	p17
Comparison jump (Variable : Numeric)	43h,23h, REG1,mode, d1,d2,[d3,d4,] rL,rH	Compare REG1 with numerical value 'd', and jump to the specified relative address if the comparison condition evaluates to true. REG1: Variable to compare mode: Comparison mode d1: Comparison value × 01h d2: Comparison value × 0100h d3: Comparison value × 010000h (if 32-bit) d4: Comparison value × 01000000h (if 32-bit) rL: Relative jump address, lower byte rH: Relative jump address, upper byte	p10	p17
Comparison jump (Variable : Variable)	43h,52h, REG1,mode,REG2, rL,rH	Compare REG1 with REG2, and jump to the specified relative address if the comparison condition evaluates to true. REG1: Variable to compare mode: Comparison mode REG2: Variable to compare rL: Relative jump address, lower byte rH: Relative jump address, upper byte	p10	p18
Unconditional jump	4Ah,50h,rL,rH	Jump to the specified relative address. rL: Relative jump address, lower byte rH: Relative jump address, upper byte	p10	p18
Subroutine call	4Ah,53h,rL,rH	Call the address of the top of subroutine. rL: Relative call address, lower byte rH: Relative call address, upper byte	p11	p18
Subroutine return	52h,54h	Subroutine ends, returning to the next address after the subroutine call instruction.	p11	p18
Event enable	45h,45h	Event (interrupt) is enabled.	p11	p19 p20
Event disable	45h,44h	Event (interrupt) is disabled.	p11	p19 p20

Instruction name	Hex Code	Operation	Details	Example
Timer event setup	45h,54h,en,rL,rH	Configure Timer event. en: Enable / Disable rL: Event routine relative call address, lower byte rH: Event routine relative call address, upper byte	p11	p19
Port event setup	45h,50h,en,mode,io, dataL,dataH, maskL,maskH, rL,rH	Configure Port event. en: Enable / Disable mode: Event generating conditions io: General purpose I/O port number dataL: Comparison data, lower byte dataH: Comparison data, upper byte maskL: Mask data, lower byte maskH: Mask data, upper byte rL: Event routine relative call address, lower byte rH: Event routine relative call address, upper byte	p12	p20
Event return	52h,45h	Event processing ends, returning to the next address from where the Event occurred.	p12	p19 p20
Wait	57h,54h	Pause Program Macro processing for a set time.	p12	p20
Variable data length set	4Dh,44h,d1	Sets the data length for Variables. d1=30h: 16-bit (2 bytes) (Default) d1=31h: 32-bit (4 bytes)	p13	p22
Memory Access data length set	57h,d1	Sets the data length for Memory Access. d1=31h: 8-bit (1 byte) (Default) d1=32h: 16-bit (2 bytes) d1=34h: 32-bit (4 bytes)	p13	p23
Memory Access area set	53h,47h,d1	Sets the applicable area for Memory Access. d1=00h: Currently-executing Program Macro (Default) d1=20h: Display Memory d1=30h: General-purpose RAM d1=31h: General-purpose FROM d1=40h: RAM Bit Image d1=41h: FROM Bit Image d1=51h: Memory SW	p14	p23
Display Memory write (Numeric)	56h,23h,REG1,d	Write pattern data 'd' to the location in Display Memory specified by REG1. REG1: Display memory address d: Pattern data	p14	p21
Display Memory write (Variable)	56h,52h,REG1,REG2	Write the contents of Variable REG2 to the location in Display Memory specified by REG1.	p14	p21
Display Memory read	76h,52h,REG1,REG2	Read data at the location in Display Memory specified by REG2 into REG1. REG1: Variable to store data REG2: Display memory address	p14	p22
Program Macro End	42h,4Bh	Program Macro ends.	p14	p15

Note: d1,d2,d3,d4: Numerical value (unsigned). rL,rH: Relative address (16-bit, signed)

Caution regarding instruction interpretation:
If the instruction code check does not correspond to a Program Macro instruction, that code is skipped and the next data is checked. For example, if the sequence P Z 05h 00h S T A R T is defined, P through A is skipped as there is no corresponding instruction, then R T is processed as Subroutine return code. This example shows how unexpected instructions can be executed, so it is important to ensure that incorrect code is not defined.

Data area usage:
Macro data memory, Display Memory, General-purpose RAM/FROM, Bit Image RAM/FROM and Memory SW can be used as the data area.
Variables a – z are used for memory access. Refer to 3.5.8 Comparison jump (Variable : Numeric) for usage example.

3.4 Instruction set details

3.4.1 Command transfer (Numeric) P # nL nH d1 d2 ... dn

Operation image: PRINT d1 d2 ... d(n)

Code: 50h 23h nL nH d1 d2 ... dn

nL: Data sequence length, lower byte
nH: Data sequence length, upper byte
d1...dn: Data sequence

Definable area: 0001h ≤ (nL + nH×100h) ≤ 00FFh

Function: Transfer data d1...dn to the Command processing unit.

The following commands cannot be defined in the data sequence:
Initialize, Macro execution, RAM Macro define / delete, User setup mode start, “US (e” group commands (FROM bit image definition, Download character save, etc), Macro end condition, Memory re-write mode.
For details on each command, refer to the “General Function” Software Specification.
This instruction can transfer several commands at once, or a single command could be transferred in portions using several instructions.

3.4.2 Command transfer (Variable) P R nL nH REG1 REG2 … REGn

Operation image: PRINT REG1 REG2 ... REGn

Code: 50h 52h nL nH REG1 REG2 ... REGn

nL: Variable sequence length, lower byte
nH: Variable sequence length, upper byte
REG1...REGn: Variable sequence (only lower 8 bits of Variable contents used)

Definable area: 0001h ≤ (nL + nH×100h) ≤ 00FFh

Function: Transfer the contents of Variables REG1...REGn to the Command processing unit.

The following commands cannot be defined in the Variable sequence:
Initialize, Macro execution, RAM Macro define / delete, User setup mode start, “US (e” group commands (FROM bit image definition, Download character save, etc), Macro end condition, Memory re-write mode.
For details on each command, refer to the “General Function” Software Specification.
This instruction can transfer several commands at once, or a single command could be transferred in portions using several instructions.

3.4.3 Arithmetic operation (Variable←Variable, Numeric) L # REG1 mode d1 d2 [d3 d4]

Operation image:
16-bit: LET REG1 = REG1 mode (d1 + d2×0100h)
32-bit: LET REG1 = REG1 mode (d1 + d2×0100h + d3×010000h + d4×01000000h)

Code: 4Ch 23h REG1 mode d1 d2 [d3 d4]

REG1: Variable to store result
mode: Operation mode
= : Assign (3Dh)
+ : Add (2Bh)
- : Subtract (2Dh)
* : Multiply (2Ah)
/ : Divide (2Fh)
% : Remainder (25h)
| : Bit-wise OR (7Ch)
& : Bit-wise AND (26h)
^ : Bit-wise XOR (5Eh)
d1: Numeric value × 01h
d2: Numeric value × 0100h
d3: Numeric value × 010000h (if 32-bit)
d4: Numeric value × 01000000h (if 32-bit)

Definable area: 0000h ≤ (d1 + d2×0100h) ≤ FFFFh
00000000h ≤ (d1 + d2×0100h + d3×010000h + d4×01000000h) ≤ FFFFFFFFh

Function: Perform calculation on REG1 with numerical value 'd' and store the result in REG1.
For mode '=', numerical value 'd' is copied to REG1.
Do not set d=0000h (or 00000000h) if mode is '/' or '%'.
If calculation result exceeds bit length, the lower effective bits only are stored into REG1.
Instruction is invalid if REG1 is read-only.

3.4.4 Shift Variable L # REG1 mode d1

Operation image: LET REG1 = REG1 mode d1

Code: 4Ch 23h REG1 mode d1

REG1: Variable

mode: Operation mode

> : Right shift (3Eh)

< : Left shift (3Ch)

d1: Shift bit count

Definable area: 00h ≤ d1 ≤ FFh (lower 5 bits only, 0 – 31 shifts)

Function: Shift the contents of REG1 by d1 bits.

Only the effective number of bits are stored into REG1.

Instruction is invalid if REG1 is read-only.

3.4.5 Increment / Decrement Variable L # REG1 mode

Operation image: LET REG1 = REG1 mode

Code: 4Ch 23h REG1 mode

REG1: Variable

mode: Operation mode

i: increment (69h)

d: decrement (64h)

Function: Increment (+1) or Decrement (−1) the contents of REG1.

Only the effective number of bits are stored into REG1.

Instruction is invalid if REG1 is read-only.

3.4.6 Arithmetic operation (Variable1←Variable1, Variable2) L R REG1 mode REG2

Operation image: LET REG1 = REG1 mode REG2

Code: 4Ch 52h REG1 mode REG2

REG1: Variable to store result

mode: Operation mode

= : Assign (3Dh)

+ : Add (2Bh)

- : Subtract (2Dh)

* : Multiply (2Ah)

/ : Divide (2Fh)

% : Remainder (25h)

| : Bit-wise OR (7Ch)

& : Bit-wise AND (26h)

^ : Bit-wise XOR (5Eh)

REG2: Operation variable 2

Function: Perform calculation on REG1 with REG2 and store the result in REG1.

For mode '=', REG2 is copied to REG1.

Do not use mode '/' or '%' if REG2 contents is 0h

If calculation result exceeds bit length, the lower effective bits only are stored into REG1.

Instruction is invalid if REG1 is read-only.

3.4.7 Push Variable P S REG1

Operation image: LET REG_STACK (SP) = REG1

LET SP + SP + 1

Code: 50h 53h REG1

REG1: Variable

Function: Store the value in Variable REG1 onto the stack.

Stack area has 32 levels – a maximum of 32 values can be stored.

If Push variable is attempted at level 32, a Program Macro error occurs.

3.4.8 Pop Variable P L REG1

Operation image: LET SP + SP − 1

LET REG1 = REG_STACK (SP)

Code: 50h 4Ch REG1

REG1: Variable

Function: Retrieve a value off the stack into REG1.

If Pop variable is attempted at stack level 0, a Program Macro error occurs.

3.4.9 Comparison jump (Variable : Numeric) C # REG1 mode d1 d2 [d3 d4] rL rH

Operation image:
16-bit: IF REG1 mode (d1 + d2×0100h) THEN GOTO (rL + rH×0100h)
32-bit: IF REG1 mode (d1 + d2×0100h + d3×010000h + d4×01000000h)
THEN GOTO (rL+rH×0100h)

Code: 43h 23h REG1 mode d1 d2 [d3 d4] rL rH

REG1: Variable to compare
mode: Comparison mode
= : Equal (3Dh)
! : Not equal (21h)
< : Less than (3Ch)
> : Greater than (3Eh)
(: Less than or equal to (28h)
) : Greater than or equal to (29h)
d1: Comparison value × 01h
d2: Comparison value × 0100h
d3: Comparison value × 010000h (if 32-bit)
d4: Comparison value × 01000000h (if 32-bit)
rL: Relative jump address, lower byte
rH: Relative jump address, upper byte

Definable area: 16-bit:
0000h ≤ (d1 + d2×100h) ≤ FFFFh
32-bit:
00000000h ≤ (d1 + d2×100h + d3×10000h + d4×1000000h) ≤ FFFFFFFFh
−32768(8000h) ≤ (rL + rH×100h) ≤ 32767(7FFFh)

Function: Compare REG1 with numerical value 'd', and jump to the specified relative address if the comparison condition evaluates to true.

3.4.10 Comparison jump (Variable : Variable) C R REG1 mode REG2 rL rH

Operation image: IF REG1 mode REG2 THEN GOTO (rL + rH×0100h)

Code: 43h 52h REG1 mode REG2 rL rH

REG1: Variable to compare
mode: Comparison mode
= : Equal (3Dh)
! : Not equal (21h)
< : Less than (3Ch)
> : Greater than (3Eh)
(: Less than or equal to (28h)
) : Greater than or equal to (29h)
REG2: Variable to compare
rL: Relative jump address, lower byte
rH: Relative jump address, upper byte

Definable area: −32768(8000h) ≤ (rL + rH×100h) ≤ 32767(7FFFh)

Function: Compare REG1 with REG2, and jump to the specified relative address if the comparison condition evaluates to true.

3.4.11 Unconditional jump

Operation image: GOTO (rL + rH×0100h)

Code: 4Ah 50h rL rH

rL: Relative jump address, lower byte
rH: Relative jump address, upper byte

Definable area: −32768(8000h) ≤ (rL + rH×100h) ≤ 32767(7FFFh)

Function: Jump to the specified relative address.

3.4.12 Subroutine call J S rL rH

Operation image: CALL (rL + rH×0100h)

Code: 4Ah 53h rL rH

rL: Relative call address, lower byte

rH: Relative call address, upper byte

Definable area: −32768(8000h) ≤ (rL + rH×100h) ≤ 32767(7FFFh)

Function: Call the address of the top of subroutine.

3.4.13 Subroutine return R T

Code: 52h 54h

Function: Subroutine ends, returning to the next address after the subroutine call instruction.

3.4.14 Event enable E E

Code: 45h 45h

Function: Event (interrupt) is enabled.

3.4.15 Event disable E D

Code: 45h 44h

Function: Event (interrupt) is disabled.

3.4.16 Timer event setup E T en rL rH

Code: 45h 54h en rL rH

en: Enable / Disable

rL: Event routine relative call address, lower byte

rH: Event routine relative call address, upper byte

Definable area: 00h ≤ en ≤ 01h

en = 00h: Disabled

en = 01h: Enabled

−32768(8000h) ≤ (rL + rH×100h) ≤ 32767(7FFFh)

Function: Configure Timer event.

If enabled (en=1), the specified relative address is called when the Timer event is generated. Event is not generated if the “Event enable” instruction has not been executed. If this instruction is called several times, only the last setup is effective. Timer event interval is set by writing to the 16-bit Variable '?' (Timer counter value) using one of the “Arithmetic operation” instructions. This sets the Timer counter value and the reload value, and re-starts the timer.

Timer counter is decremented by one at intervals of approximately 14ms. If the Timer counter is 0, an Event request is generated and the configured Event routine is called. The timer is immediately restarted and continues, and will again generate an Event request when it becomes 0. To stop Timer events, set Disabled (en=0). Writing 0 to '?' has the same effect as writing '1' and will result in an interval of approximately 14ms.

Caution: If the Timer event interval is shorter than the execution time of the Event routine, the Event routine will execute continuously in an endless loop.

Note: The 14ms intervals referred to above are for the 256×64 dot type VFD module. The interval varies depending on the VFD module. Refer to the VFD Module model-specific Timing Unit listed in the “General Function” Software Specification.

3.4.17 Port event setup E P en mode io dataL dataH maskL maskH rL rH

Code: 45h 50h en mode io dataL dataH maskL maskH rL rH

en: Enable / Disable
mode: Event generating conditions
io: General purpose I/O port number
dataL: Comparison data, lower byte
dataH: Comparison data, upper byte
maskL: Mask data, lower byte
maskH: Mask data, upper byte
rL: Event routine relative call address, lower byte
rH: Event routine relative call address, upper byte

Definable area: 00h ≤ en ≤ 01h
en = 00h: Disabled, en = 01h: Enabled
00h ≤ mode ≤ 01h
mode = 00h: Event generation on match
mode = 01h: Event generation on mismatch
30h ≤ io ≤ 31h
io = 30h: 8-bit port (Port 0), io = 31h: 4-bit port (Port 1)
0000h ≤ (dataL + dataH×0100h) ≤ FFFFh (only lower 8 bits used)
0000h ≤ (maskL + maskH×0100h) ≤ FFFFh (only lower 8 bits used)
−32768(8000h) ≤ (rL + rH×100h) ≤ 32767(7FFFh)

Function: Configure Port event.

If enabled (en=1), the specified relative address is called when the event is generated.
Event is not generated if the “Event enable” instruction has not been executed.
If this instruction is called several times, only the last setup is effective.
Port event generating conditions:
mode=00h: Event is generated when (io & mask == data) evaluates to true (match).
mode=01h: Event is generated when (io & mask != data) evaluates to true (mismatch).
However, this is disabled during execution of event processing or command transfer, so the input signal to the port should have regard to the required time for this.

3.4.18 Event return R E

Code: 52h 45h

Function: Event processing ends, returning to the next address from where the Event occurred.

3.4.19 Wait W T

Code: 57h 54h

Function: Pause Program Macro processing for a set time.

Depending on internal timing, pause time is 14 – 28ms.

Note: The 14 – 28ms interval referred to above is for the 256×64 dot type VFD module. The internal timing varies depending on the VFD module. Pause time is 1 – 2 times the applicable Timing Unit specified in the “General Function” Software Specification.

3.4.20 Variable data length set M D d1

Code: 4Dh 44h d1

d1:Data length

Definable area: d1=30h: 16-bit (2 bytes)
d1=31h: 32-bit (4 bytes)

Default: d1=30h: 16-bit (2 bytes)

Function: Sets the data length for Variables.

3.4.21 Memory Access data length set W d1

Code: 57h d1

d1: Data length

Definable area: d1 = 31h, 32h, 34h
d1=31h: 8-bit (1 byte)
d1=32h: 16-bit (2 bytes)
d1=34h: 32-bit (4 bytes)

Default: d1=31h: 8-bit (1 byte)

Function: Sets the data length for Memory Access.

When writing to memory, only the specified data length is written.

When reading, only the specified data length is read, and any unread upper bits are set to 00h.

Memory write example:

Variable x

b31 -- b24	b23 -- b16	b15 -- b8	b7 -- b0
REGx(d4)	REGx(d3)	REGx(d2)	REGx(d1)

↓ Memory write

Memory

	Memory address						
	0	1	2	3	4	5	6
8-bit		REGx(d1)					
16-bit		REGx(d1)	REGx(d2)				
32-bit		REGx(d1)	REGx(d2)	REGx(d3)	REGx(d4)		

Memory read example:

Memory

	Memory address						
	0	1	2	3	4	5	6
8-bit		REGx(d1)					
16-bit		REGx(d1)	REGx(d2)				
32-bit		REGx(d1)	REGx(d2)	REGx(d3)	REGx(d4)		

↓ Memory read

Variable x

	b31 -- b24	b23 -- b16	b15 -- b8	b7 -- b0
8-bit	00h	00h	00h	REGx(d1)
16-bit	00h	00h	REGx(d2)	REGx(d1)
32-bit	REGx(d4)	REGx(d3)	REGx(d2)	REGx(d1)

3.4.22 Memory Access area set S G d1

Code: 53h 47h d1

d1: Memory area

Definable area: d1 = 00h, 20h, 30h, 31h, 40h, 41h, 51h

d1=00h: Currently-executing Program Macro (Default)
d1=20h: Display Memory
d1=30h: General-purpose RAM
d1=31h: General-purpose FROM
d1=40h: RAM Bit Image
d1=41h: FROM Bit Image
d1=51h: Memory SW

Default: d1 = 00h (Currently-executing Program Macro area)

Function: Sets the applicable area for Memory Access.

d1	Memory area	Read	Write
00h	Currently-executing Program Macro area	○	○ (RAM Macro only)
20h	Display Memory (VRAM)	○	○
30h	General-purpose RAM	○	○
31h	General-purpose FROM	○	×
40h	RAM Bit Image	○	○
41h	FROM Bit Image	○	×
51h	Memory SW	○	×

3.4.23 Display Memory write (Numeric) V # REG1 d

Operation image: Vram[REG1 & 0FFFh] = d

(Note: 0FFFh is the Display Memory final address value, which varies for different VFD modules)

Code: 56h 23h REG1 d

REG1: Display Memory address

d: Pattern data

Definable area: 00h ≤ d ≤ FFh

Function: Write pattern data 'd' to the location in Display Memory specified by REG1.

Refer to “General Function” Software Specification, “Display Memory” section for Display Memory details.

3.4.24 Display Memory write (Variable) V R REG1 REG2

Operation image: Vram[REG1 & 0FFFh] = REG2

(Note: 0FFFh is the Display Memory final address value, which varies for different VFD modules)

Code: 56h 52h REG1 REG2

REG1: Display Memory address

REG2: Pattern data (only lower 8 bits used)

Definable area: 00h ≤ REG2 ≤ FFh

Function: Write contents of REG2 (pattern data) to the location in Display Memory specified by REG1.

Refer to “General Function” Software Specification, “Display Memory” section for Display Memory details.

3.4.25 Display Memory read v R REG1 REG2

Operation image: REG1 = Vram[REG2 & 0FFFh]

(Note: 0FFFh is the Display Memory final address value, which varies for different VFD modules)

Code: 76h 52h REG1 REG2

REG1: Variable to store data (only lower 8 bits used)

REG2: Display Memory address

Function: Read data at the location in Display Memory specified by REG2 into REG1.

Only the lower 8 bits are read. Upper bits are set to 00h.

Refer to “General Function” Software Specification, “Display Memory” section for Display Memory details.

3.4.26 Program Macro End B K

Code: 42h 4Bh

Function: Program Macro ends.

3.5 Programming Examples

The sample source in this section shows each instruction separately, however the data for a Program Macro is defined as a continuous block.
The Program Macro that is defined is stored in the Macro Data memory, with the first definition data located at address 0000h. The address referred to by Variables a – z is an absolute address, based at 0000h in Macro Data memory.
Addresses specified by “Call” or “Jump” instructions are relative addresses, calculated from the next code byte after “rH”.
The below table shows an example of actual code and relative address values.
In this example, the 'P' immediately after the jump instruction is at the relative address 0000h, and so a jump to 'L' is possible by setting rL=05h, rH=00h in the Unconditional jump instruction (J P rL rH).

Relative address 'r'	FFFCh (−4)	FFFDh (−3)	FFFEh (−2)	FFFFh (−1)	0000h (0)	0001h (1)	0002h (2)	0003h (3)	0004h (4)	0005h (5)	0006h (6)	0007h (7)	
Code (example)	J	P	rL	rH	P	R	nL	nH	A	L	#	A	...

Note: Numbers in () are decimal numbers.
In 3.5.2 Command transfer (Variable), the applicable relative address values are shown.

3.5.1 Command transfer (Numeric)

(a) Display “itron”. (Definition data: 11 bytes)

Data	50h	23h	05h	00h	69h	74h	72h	6Fh	6Eh	Display “itron”.
Code	P	#	nL	nH	i	t	r	o	n	
Data	42h	4Bh	End Program Macro.							
Code	B	K								

(b) Curtain display action command. (Definition data: 13 bytes)

Data	50h	23h	07h	00h	1Fh	28h	61h	12h	02h	01h	FFh	Curtain action display command.
Code	P	#	nL	nH	US	(	a	n	v	s	p	
Data	42h	4Bh	End Program Macro.									
Code	B	K										

Curtain display action command

3.5.2 Command transfer (Variable)

Execute Curtain display action command twice, changing the contents of parameter 'p' (curtain pattern).
(Definition data: 37 bytes)

Relative address	FFEDh	FFDEh	FFDFh	FFE0h	FFE1h	FFE2h	Load 33h into Variable A.
Data	4Ch	23h	41h	3Dh	33h	00h	
Code	L	#	A	=	d1	d2	

Relative address	FFE3h	FFE4h	FFE5h	FFE6h	FFE7h	FFE8h	FFE9h	FFEAh	FFEBh	FFECh	By executing these two instructions, Curtain display action is executed.
Data	50h	23h	06h	00h	1Fh	28h	61h	12h	02h	01h	
Code	P	#	nL	nH	US	(	a	n	v	s	
Relative address	FFEDh	FFEEh	FFFFh	FFF0h	FFF1h	(A: Parameter p)					
Data	50h	52h	01h	00h	41h						
Code	P	R	nL	nH	A						

Relative address	FFF2h	FFF3h	FFF4h	FFF5h	FFF6h	FFF7h	Store result of A + 99h into Variable 'A'.
Data	4Ch	23h	41h	2Bh	99h	00h	
Code	L	#	A	+	d1	d2	

Relative address	FFF8h	FFF9h	FFFAh	FFFBh	FFFCh	FFFDh	FFFEh	FFFFh	If contents of Variable 'A' is less than or equal to CCh, jump to the specified address. (Jump to P(50h))
Data	43h	23h	41h	28h	CCh	00h	E3h	FFh	
Code	C	#	A	(	d1	d2	rL	rH	

Relative address	0000h	0001h	
Data	42h	4Bh	End Program Macro.
Code	B	K	

Curtain display action section

3.5.3 Arithmetic operation (Variable←Variable, Numeric)

Store 20h into Variable 'A', add 40h to 'A', then display contents of 'A'. (Definition data: 19 bytes)

Data	4Ch	23h	41h	3Dh	20h	00h	Store 20h into Variable 'A'.
Code	L	#	A	=	d1	d2	
Data	4Ch	23h	41h	2Bh	40h	00h	Store result of A + 40h into Variable 'A'.
Code	L	#	A	+	d1	d2	
Data	50h	52h	01h	00h	41h		Display contents of Variable 'A' (60h = '`').
Code	P	R	nL	nH	A		
Data	42h	4Bh					End Program Macro.
Code	B	K					

3.5.4 Shift Variable

Store 12h into Variable 'A', left shift twice, then display contents of 'A'. (Definition data: 18 bytes)

Data	4Ch	23h	41h	3Dh	12h	00h	Store 12h into Variable 'A'.
Code	L	#	A	=	d1	d2	
Data	4Ch	23h	41h	3Ch	02h		Left shift Variable 'A' by two bits.
Code	L	#	A	<	d1		
Data	50h	52h	01h	00h	41h		Display contents of Variable 'A' (48h = 'H').
Code	P	R	nL	nH	A		
Data	42h	4Bh					End Program Macro.
Code	B	K					

3.5.5 Increment / Decrement Variable

Store 41h into Variable 'A', increment, then display contents of 'A'. (Definition data: 17 bytes)

Data	4Ch	23h	41h	3Dh	41h	00h	Store 41h into Variable 'A'.
Code	L	#	A	=	d1	d2	
Data	4Ch	23h	41h	69h			Increment Variable 'A'.
Code	L	#	A	i			
Data	50h	52h	01h	00h	41h		Display contents of Variable 'A' (42h = 'B').
Code	P	R	nL	nH	A		
Data	42h	4Bh					End Program Macro.
Code	B	K					

3.5.6 Arithmetic operation (Variable1←Variable1, Variable2)

Store 50h into Variable 'A' and 20h into Variable 'B', subtract B from A, then display contents of 'A'.

Data	4Ch	23h	41h	3Dh	50h	00h	Store 50h into Variable 'A'.
Code	L	#	A	=	d1	d2	
Data	4Ch	23h	42h	3Dh	20h	00h	Store 20h into Variable 'B'.
Code	L	#	B	=	d1	d2	
Data	4Ch	52h	41h	2Dh	42h		Store the result of A − B into Variable 'A'.
Code	L	R	A	-	B		
Data	50h	52h	01h	00h	41h		Display contents of Variable 'A' (30h = '0').
Code	P	R	nL	nH	A		
Data	42h	4Bh					End Program Macro.
Code	B	K					

3.5.7 Push / Pop Variable

Push Variable 'A' then display contents of A after Pop. (Definition data: 30 bytes)

Data	4Ch	23h	41h	3Dh	30h	00h	Store 30h into Variable 'A'.
Code	L	#	A	=	d1	d2	
Data	50h	53h	41h	Push Variable 'A'.			
Code	P	S	A				
Data	4Ch	23h	41h	3Dh	31h	00h	Store 31h into Variable 'A'.
Code	L	#	A	=	d1	d2	
Data	50h	52h	01h	00h	41h	Display contents of Variable 'A' (31h = '1').	
Code	P	R	nL	nH	A		
Data	50h	4Ch	41h	Pop Variable 'A'.			
Code	P	L	A				
Data	50h	52h	01h	00h	41h	Display contents of Variable 'A' (30h = '0').	
Code	P	R	nL	nH	A		
Data	42h	4Bh	End Program Macro.				
Code	B	K					

3.5.8 Comparison jump (Variable : Numeric)

Progressively read and display, one character at a time “Good Morning!” stored in the data area. (Definition data: 49 bytes)

Data	4Ah	50h	0Dh	00h	Jump to the specified address. (Jump to L (4Ch))									
Code	J	P	rL	rH										
Data	47h	6Fh	6Fh	64h	20h	4Dh	6Fh	72h	6Eh	69h	6Eh	67h	21h	Define “Good Morning!”.
Code	G	o	o	d		M	o	r	n	i	n	g	!	
Data	4Ch	23h	41h	3Dh	04h	00h	Store 04h into Variable 'A'.							
Code	L	#	A	=	d1	d2								
Data	4Ch	52h	42h	3Dh	61h	Store the data at at the Macro Data memory address indicated by variable 'A' into Variable 'B'.								
Code	L	R	B	=	a									
Data	50h	52h	01h	00h	42h	Display contents of Variable 'B'.								
Code	P	R	nL	nH	B									
Data	4Ch	23h	41h	2Bh	01h	00h	Store result of A + 01h into Variable 'A'.							
Code	L	#	A	+	d1	d2								
Data	43h	23h	41h	28h	10h	00h	E8h	FFh	Jump to the specified address if Variable 'A' is less than or equal to 10h. (Jump to L (4Ch))					
Code	C	#	A	(	d1	d2	rL	rH						
Data	42h	4Bh	End Program Macro.											
Code	B	K												

Data definition area

Explanation: 'J' at line 1 is located at 0000h in Macro data memory, and 'G' at line 2 is located at 0004h. Accordingly, setting 0004h into Variable 'A' (line 3), and reading by Variable 'a', the data 'G' at location 0004h is stored into Variable 'B' (line 4).

3.5.9 Comparison jump (Variable : Variable)

Display Variable 'A' (initial value: 20h) while incrementing by one until it is equal to Variable 'B' (30h).
(Definition data: 32 bytes)

Data	4Ch	23h	41h	3Dh	20h	00h	Store 20h into Variable 'A'.
Code	L	#	A	=	d1	d2	
Data	4Ch	23h	42h	3Dh	30h	00h	Store 30h into Variable 'B'.
Code	L	#	B	=	d1	d2	
Data	50h	52h	01h	00h	41h		Display contents of Variable 'A'.
Code	P	R	nL	nH	A		
Data	4Ch	23h	41h	2Bh	01h	00h	Store result of A + 01h into Variable 'A'.
Code	L	#	A	+	d1	d2	
Data	43h	52h	41h	3Ch	42h	EEh	FFh	Jump to the specified address if A is less than B. (Jump to P (50h))
Code	C	R	A	<	B	rL	rH	
Data	42h	4Bh					End Program Macro.
Code	B	K					

3.5.10 Unconditional jump

Read 1 byte from the receive buffer, and if it is other than FFh, display it + '/'.
If it is FFh, end Program Macro. Repeat this sequence. (Definition data: 29 bytes)

Data	4Ch	52h	41h	3Dh	40h				Read receive buffer byte into Variable 'A'.
Code	L	R	A	=	@				
Data	50h	52h	01h	00h	41h				Display contents of Variable 'A'.
Code	P	R	nL	nH	A				
Data	43h	23h	41h	3Dh	FFh	00h	09h	00h	Jump to specified address if Variable 'A' is FFh. (Jump to B (42h))
Code	C	#	A	=	d1	d2	rL	rH	
Data	50h	23h	01h	00h	2Fh				Display '/'.
Code	P	#	nL	nH	/				
Data	4Ah	50h	E5h	FFh					Jump to specified address. (Jump to L(4Ch))
Code	J	P	rL	rH					
Data	42h	4Bh							End Program Macro.
Code	B	K							

Example: While Program Macro is running,

Send “ABC” → “A/B/C/” is displayed
Send FFh → Program Macro ends

3.5.11 Subroutine call, Subroutine return

Display contents of Variable 'A' after calling a subroutine. (Definition data: 25 bytes)

Data	4Ch	23h	41h	3Dh	20h	00h	Store 20h into Variable 'A'.
Code	L	#	A	=	d1	d2	
Data	4Ah	53h	07h	00h			Call subroutine at specified address.
Code	J	S	rL	rH			(Jump to L (4Ch))
Data	50h	52h	01h	00h	41h		Display contents of Variable 'A' (21h = '!').
Code	P	R	nL	nH	A		
Data	42h	4Bh					End Program Macro.
Code	B	K					
Data	4Ch	23h	41h	2Bh	01h	00h	Store result of A + 01h into Variable 'A'.
Code	L	#	A	+	d1	d2	
Data	52h	54h					End subroutine.
Code	R	T					

Subroutine

3.5.12 Timer event

Upon Timer event generation, if there is data in the receive buffer, display “END” and end Program Macro.
(Definition data: 61 bytes)

Data	4Ch	23h	41h	3Dh	00h	00h	Store 00h into Variable 'A'.
Code	L	#	A	=	d1	d2	
Data	4Ch	23h	3Fh	3Dh	10h	00h	Set Timer counter value to 10h.
Code	L	#	?	=	d1	d2	
Data	45h	54h	01h	06h	00h		Configure Timer event to execute from the specified location when it occurs. (Execute from C (43h))
Code	E	T	en	rL	rH		
Data	45h	45h	Event enable.				
Code	E	E					
Data	4Ah	50h	10h	00h			Jump to the specified address. (Jump to P (50h))
Code	J	P	rL	rH			

Data	43h	23h	25h	3Dh	00h	00h	06h	00h
Code	C	#	%	=	d1	d2	rL	rH
Data	4Ch	23h	41h	3Dh	01h	00h		
Code	L	#	A	=	d1	d2		
Data	52h	45h						
Code	R	E						

Determine if there is data in the receive buffer.

Store 01h into Variable 'A'. (Yes)

End event. (No)

Data	50h	23h	01h	00h	21h			
Code	P	#	nL	nH	!			
Data	43h	23h	41h	21h	01h	00h	F3h	FFh
Code	C	#	A	!	d1	d2	rL	rH
Data	50h	23h	03h	00h	45h	4Eh	44h	
Code	P	#	nL	nH	E	N	D	
Data	42h	4Bh						
Code	B	K						

If Variable 'A' is not 01h, continue displaying '!'.
If Variable 'A' is 01h, display “END” and end Program Macro.

Timer event routine

Normal program

3.5.13 Port event

When Port event is generated, store 01h into Variable 'A', display “END” and end Program Macro.
(Definition data: 53 bytes)

Data	4Ch	23h	41h	3Dh	00h	00h	Store 00h into Variable 'A'.					
Code	L	#	A	=	d1	d2						
Data	45h	50h	01h	00h	30h	00h	00h	80h	00h	06h	00h	Configure Port event to execute from the specified location when it occurs.
Code	E	P	en	mode	io	dataL	dataH	maskL	maskH	rL	rH	
Data	45h	45h	Event enable									
Code	E	E										
Data	4Ah	50h	08h	00h	Jump to the specified address. (Jump to P (50h))							
Code	J	P	rL	rH								
Data	4Ch	23h	41h	3Dh	01h	00h	Store 01h into Variable 'A'.					
Code	L	#	A	=	d1	d2						
Data	52h	45h	End event.									
Code	R	E										

Data	50h	23h	01h	00h	21h	If Variable 'A' is not 01h, continue displaying '!'. If Variable 'A' is 01h, display “END” and end Program Macro.		
Code	P	#	nL	nH	!			
Data	43h	23h	41h	21h	01h	00h	F3h	FFh
Code	C	#	A	!	d1	d2	rL	rH
Data	50h	23h	03h	00h	45h	4Eh	44h	
Code	P	#	nL	nH	E	N	D	
Data	42h	4Bh						
Code	B	K						

Port event routine Normal program

Port event generation

In the above example, Port event is configured as mode=00h, io=30h, data=0000h, mask=0080h.
Accordingly, operation is as follows:

(1) Port 0 state	0xxxxxxx	1xxxxxxx
	&	&
(2) Mask data (mask)	10000000	10000000
(3) Result ((1) & (2))	00000000	10000000
(4) Compariosn data (data)	00000000	00000000
(3) : (4)	Match	No match
Match event generation (mode=00h)	○	×

If bit 7 of Port 0 changes to 0 (low), the instruction located at the position specified by the Port event setup instruction is executed.

3.5.14 Wait

Wait 40 times (28h) between displaying 'C' and displaying 'D'. (Definition data: 34 bytes)

Data	50h	23h	01h	00h	43h	Display 'C'.			
Code	P	#	nL	nH	C				
Data	4Ch	23h	41h	3Dh	00h	00h	Store 00h into Variable 'A'.		
Code	L	#	A	=	d1	d2			
Data	4Ch	23h	41h	2Bh	01h	00h	Store the result of A + 01h into Variable 'A'.		
Code	L	#	A	+	d1	d2			
Data	57h	54h	Wait instruction.						
Code	W	T							
Data	43h	23h	41h	21h	28h	00h	F0h	FFh	Jump to the specified address if Variable 'A' is not 28h. (Jump to L (4Ch))
Code	C	#	A	!	d1	d2	rL	rH	
Data	50h	23h	01h	00h	44h	Display 'D'.			
Code	P	#	nL	nH	D				
Data	42h	4Bh	End Program Macro.						
Code	B	K							

Execute Wait 40 times

3.5.15 Display Memory write (Numeric)

Display a 6×8 dot solid square at top left of display. (Definition data: 26 bytes)

Data	4Ch	23h	41h	3Dh	00h	00h		Store 00h into Variable 'A'.
Code	L	#	A	=	d1	d2		
Data	56h	23h	41h	FFh				Write FFh to Display Memory address indicated by Variable 'A'.
Code	V	#	A	d				
Data	4Ch	23h	41h	2Bh	08h	00h		Store result of A + 08h into Variable 'A'.
Code	L	#	A	+	d1	d2		
Data	43h	23h	41h	28h	28h	00h	EEh	FFh	Jump to specified address if Variable 'A' is less than or equal to 28h. (Jump to V (56h))
Code	C	#	A	(	d1	d2	rL	rH	
Data	42h	4Bh							End Program Macro.
Code	B	K							

The pattern FFh is written six times while changing the Display Memory address (REG1).

In this example, the value d1 at line 3 and line 4 needs to be set appropriately for the display screen Y-size:

Value of d1 at line 3 = Y (vertical dot count) / 8
Value of d1 at line 4 = Value of d1 at line 3 × 5

Example: 256×128 dot:

Value of d1 at line 3 = 128 / 8 → 10h
Value of d1 at line 4 = 10h × 5 → 50h

3.5.16 Display Memory write (Variable)

Write patterns 00h to 0Fh in horizontal direction from the top left of the display screen.
(Definition data: 38 bytes)

Data	4Ch	23h	41h	3Dh	00h	00h		Store 00h into Variable 'A'.
Code	L	#	A	=	d1	d2		
Data	4Ch	23h	42h	3Dh	00h	00h		Store 00h into Variable 'B'.
Code	L	#	B	=	d1	d2		
Data	56h	52h	41h	42h				Write the pattern in Variable 'B' to the Display Memory address indicated by Variable 'A'.
Code	V	R	A	B				
Data	4Ch	23h	41h	2Bh	08h	00h		Store the result of A + 08h into Variable 'A'.
Code	L	#	A	+	d1	d2		
Data	4Ch	23h	42h	2Bh	01h	00h		Store the result of B + 01h into Variable 'B'.
Code	L	#	B	+	d1	d2		
Data	43h	23h	41h	28h	78h	00h	E8h	FFh	Jump to the specified address if Variable 'A' is less than or equal to 78h. (Jump to V (56h))
Code	C	#	A	(	d1	d2	rL	rH	
Data	42h	4Bh							End Program Macro.
Code	B	K							

Variable 'A' is the Display Memory address location and Variable 'B' is the pattern data. Display Memory is written to 16 times while changing the address (line 4) and changing the pattern (line 5).

In this example, the value d1 at line 4 and line 6 needs to be set appropriately for the display screen Y-size:

Value of d1 at line 4 = Y (vertical dot count) / 8
Value of d1 at line 6 = Value of d1 at line 4 × 15

Example: 256×128 dot:

Value of d1 at line 4 = 128 / 8 → 10h
Value of d1 at line 6 = 10h × 15(0Fh) → F0h

3.5.17 Display Memory read

Display 'A' at top left of display screen, then copy it to one lower line. (Definition data: 47 bytes)

Data	50h	23h	01h	00h	41h				Display 'A'.
Code	P	#	nL	nH	A				
Data	4Ch	23h	41h	3Dh	00h	00h			Store 00h into Variable 'A'.
Code	L	#	A	=	d1	d2			
Data	4Ch	23h	43h	3Dh	01h	00h			Store 01h into Variable 'C'.
Code	L	#	C	=	d1	d2			
Data	76h	52h	42h	41h	Store the pattern data at Display Memory address indicated by Variable 'A' into Variable 'B'.				
Code	v	R	B	A					
Data	56h	52h	43h	42h	Write the pattern in Variable 'B' to the Display Memory address indicated by Variable 'C'.				
Code	V	R	C	B					
Data	4Ch	23h	41h	2Bh	08h	00h			Store result of A + 08h into Variable 'A'.
Code	L	#	A	+	d1	d2			
Data	4Ch	23h	43h	2Bh	08h	00h			Store result of C + 08h into Variable 'C'.
Code	L	#	C	+	d1	d2			
Data	43h	23h	41h	28h	28h	00h	E4h	FFh	Jump to the specified address if Variable 'A' is less than or equal to 28h. (Jump to v (76h))
Code	C	#	A	(	d1	d2	rL	rH	
Data	42h	4Bh	End Program Macro.						
Code	B	K							

Using the Display Memory read instruction, the displayed 'A' is read in 1×8-dot parts, then displayed using the Display Memory write instruction. Incrementing the read address value to the right by one column, this is repeated six times.

In this example, the value d1 at line 7, line 8, and line 9 needs to be set appropriately for the display screen Y-size:

Value of d1 at line 7 = Y (vertical dot count) / 8
Value of d1 at line 8 = Y (vertical dot count) / 8
Value of d1 at line 9 = Value of d1 at line 7 × 5

Example: 256×128 dot:

Value of d1 at line 7 = 128 / 8 → 10h
Value of d1 at line 8 = 128 / 8 → 10h
Value of d1 at line 9 = 10h × 5 → 50h

3.5.18 Variable data length set

Set Variable data length to 32 bits (4 bytes), then display contents of Variable 'A'. (Definition data: 48 bytes)

Data	4Dh	44h	31h	Set Variable data length to 32 bits (4 bytes).					
Code	M	D	d1						
Data	4Ch	23h	41h	3Dh	31h	32h	33h	34h	Store 34333231h into Variable 'A'.
Code	L	#	A	=	d1	d2	d3	d4	
Data	50h	52h	01h	00h	41h				Display lower 8 bits of Variable 'A' (31h = '1').
Code	P	R	nL	nH	A				
Data	4Ch	23h	41h	3Eh	08h				Shift Variable 'A' right by 8 bits.
Code	L	#	A	>	d1				
Data	50h	52h	01h	00h	41h				Display lower 8 bits of Variable 'A' (32h = '2').
Code	P	R	nL	nH	A				
Data	4Ch	23h	41h	3Eh	08h				Shift Variable 'A' right by 8 bits.
Code	L	#	A	>	d1				
Data	50h	52h	01h	00h	41h				Display lower 8 bits of Variable 'A' (33h = '3').
Code	P	R	nL	nH	A				
Data	4Ch	23h	41h	3Eh	08h				Shift Variable 'A' right by 8 bits.
Code	L	#	A	>	d1				
Data	50h	52h	01h	00h	41h				Display lower 8 bits of Variable 'A' (34h = '4').
Code	P	R	nL	nH	A				
Data	42h	4Bh	End Program Macro.						
Code	B	K							

3.5.19 Memory access area set

Transmit data in Memory SW and Program Macro. (Definition data: 30 bytes)

Data	53h	47h	51h	Set Memory access area to Memory SW.			
Code	S	G	d1				
Data	4Ch	23h	41h	3Dh	04h	00h	Store 0004h into Variable 'A'
Code	L	#	A	=	d1	d2	
Data	4Ch	52h	40h	3Dh	61h		Read data in Memory SW memory location indicated by Variable 'A' and transmit the data.
Code	L	R	@	=	a		
Data	53h	47h	00h	Set Memory access area to Program Macro memory.			
Code	S	G	d1				
Data	4Ch	23h	41h	3Dh	00h	00h	Store 0000h into Variable 'A'
Code	L	#	A	=	d1	d2	
Data	4Ch	52h	40h	3Dh	61h		Read data in Program Macro memory location indicated by Variable 'A' (the first byte: 53h) and transmit the data.
Code	L	R	@	=	a		
Data	42h	4Bh	End Program Macro.				
Code	B	K					

3.5.20 Memory access data length set

Display the 4-byte contents of Variable 'A' from Display Memory 00000000h, wait 2 seconds, then display 2-byte contents of Variable 'A' from Display Memory 00000001h. (Definition data: 63 bytes)

Data	4Dh	44h	31h	Set Variable data length to 32 bits (4 bytes).					
Code	M	D	1						
Data	53h	47h	20h	Set Memory access area to Display Memory.					
Code	S	G	d1						
Data	4Ch	23h	41h	3Dh	01h	02h	03h	04h	Store 04030201h into Variable 'A'.
Code	L	#	A	=	d1	d2	d3	d4	
Data	4Ch	23h	42h	3Dh	00h	00h	00h	00h	Store 00000000h into Variable 'B'.
Code	L	#	B	=	d1	d2	d3	d4	
Data	57h	34h	Set Memory access data length to 32 bits (4 bytes).						
Code	W	4							
Data	4Ch	52h	62h	3Dh	41h				Write the 32-bit (4 byte) contents of Variable 'A' to the Display Memory indicated by Variable 'B'.
Code	L	R	b	=	A				
Data	50h	23h	05h	00h	1Fh	28h	61h	01h	04h	Wait command (2 seconds).
Code	P	#	nL	nH	US	(	a	n	t	
Data	4Ch	23h	41h	3Dh	FFh	00h	CCh	33h	Store 33CC00FFh into Variable 'A'.
Code	L	#	A	=	d1	d2	d3	d4	
Data	4Ch	23h	42h	3Dh	01h	00h	00h	00h	Store 00000001h into Variable 'B'.
Code	L	#	B	=	d1	d2	d3	d4	
Data	57h	32h	Set Memory access data length to 16 bits (2 bytes).						
Code	W	2							
Data	4Ch	52h	62h	3Dh	41h				Write 16-bits (2 bytes) of contents of Variable 'A' to the Display Memory indicated by Variable 'B'.
Code	L	R	b	=	A				
Data	42h	4Bh	End Program Macro.						
Code	B	K							

Display Memory 00000000h=01h
Display Memory 00000001h=02h
Display Memory 00000002h=03h
Display Memory 00000003h=04h

Display Memory 00000000h=01h
Display Memory 00000001h=FFh
Display Memory 00000002h=00h
Display Memory 00000003h=04h

4 Restrictions

Subroutine calls can be nested up to eight levels deep. Handling a generated event counts as one level.

The following error cases cause the Program Macro to terminate (“Program Macro Error!” is displayed):

- Subroutine (including event handler) call depth exceeds eight.
- RT or RE instruction when not in a subroutine or event handler.
- Push Variable when at stack level 32.
- Pop Variable when at stack level 0.

Revision history

Specification number	Date	Revision
DS-1600-0006-00	Oct. 22, 2010	Initial release
DS-1600-0006-01	Oct. 27, 2010	Applicable models specification corrected: GU-39xxB → GU-3900B 3.4.16 3.4.19 Reference to “Short Wait” command changed to “Timing Unit” in common “General Function” Software Specification.