



for a greener tomorrow

Personal Computer Embedded Type Servo System Controller CC-Link IE Simple Motion Board/ MELSOFT EM Software Development Kit

August 2016

New Product Release
SV1608-3E

Simple Motion Board MR-EM340GF

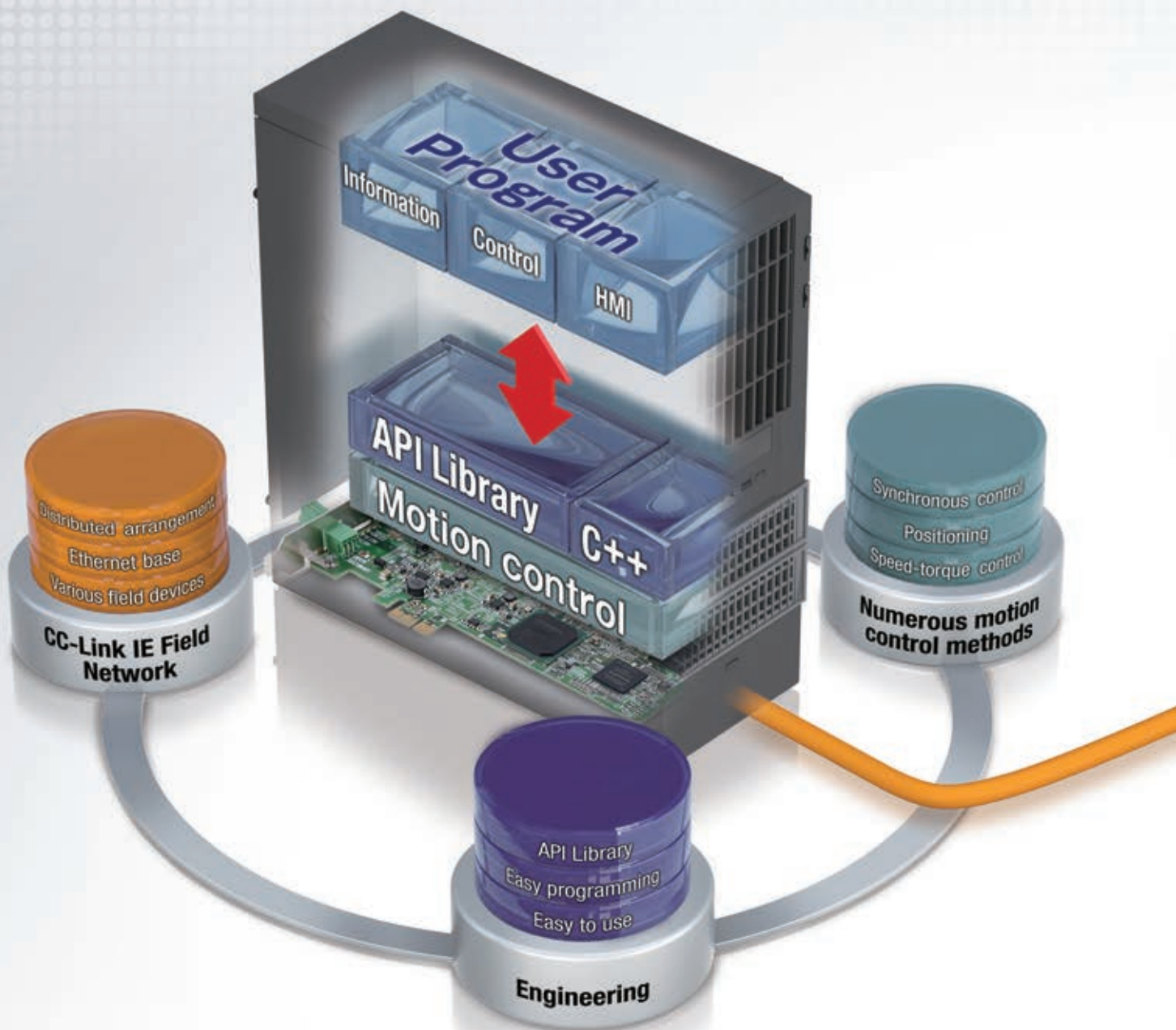


Motion Control on PC Environment with CC-Link IE Field Network

- The optimal solution with numerous motion control functions
- The satisfying development and debugging environments with MELSOFT EM Software Development Kit
- Flexible connection for various field devices with CC-Link IE Field Network
- Easy programming with Visual C++®

Personal Computer Embedded Type Servo System Controller

CC-Link IE Simple Motion Board





CC-Link IE

Numerous motion control functions on PC environment

Numerous motion control functions are available and can be applied to various machines by the Simple Motion board being embedded to an industrial personal computer (IPC) which performs customer data processing (recipe data and logging data) and image data processing.

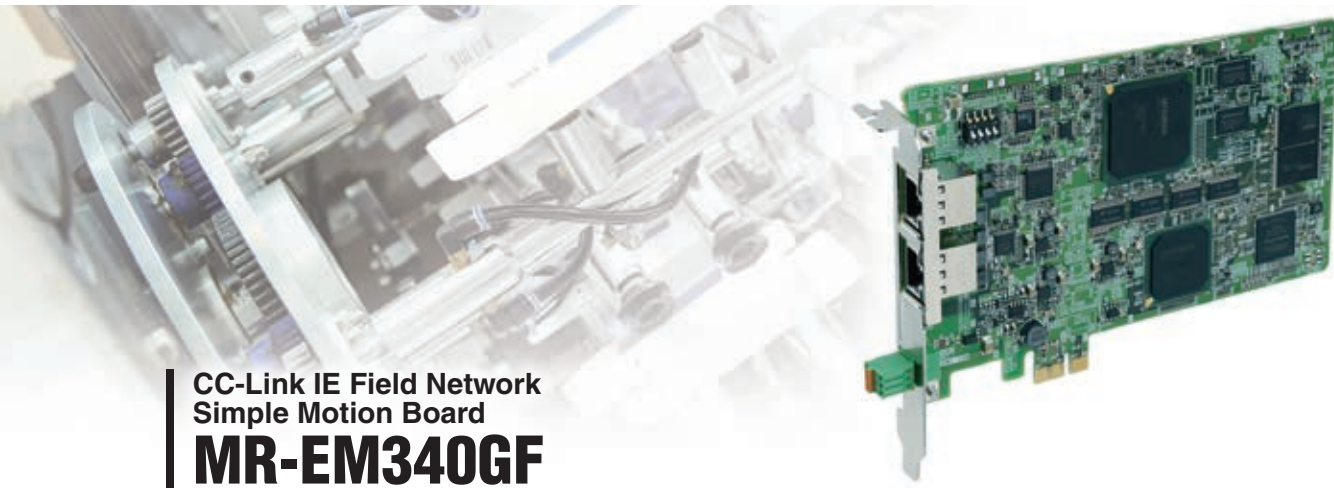
Easy programming and increased efficiency in debugging with engineering software

Easy programming is possible with Visual C++® by adding an API library and a PCI Express® device driver to the IPC. Additionally, the engineering software achieves increased efficiency in debugging because the software enables settings and monitoring of servo amplifiers and various field devices.

Seamless integration of Mitsubishi Electric's servo system into one network

CC-Link IE Field Network is a single network which combines the versatility of Ethernet and highly accurate synchronous operation for motion control. With the single network, various field devices, such as servo amplifiers, I/O modules, and high-speed counter modules, are connected with no restriction.

Simple Motion Board



CC-Link IE Field Network Simple Motion Board **MR-EM340GF**

Numerous motion control functions, such as positioning, synchronous control, and speed-torque control are performed by the Simple Motion board being embedded in a PC which supports PCI Express®.

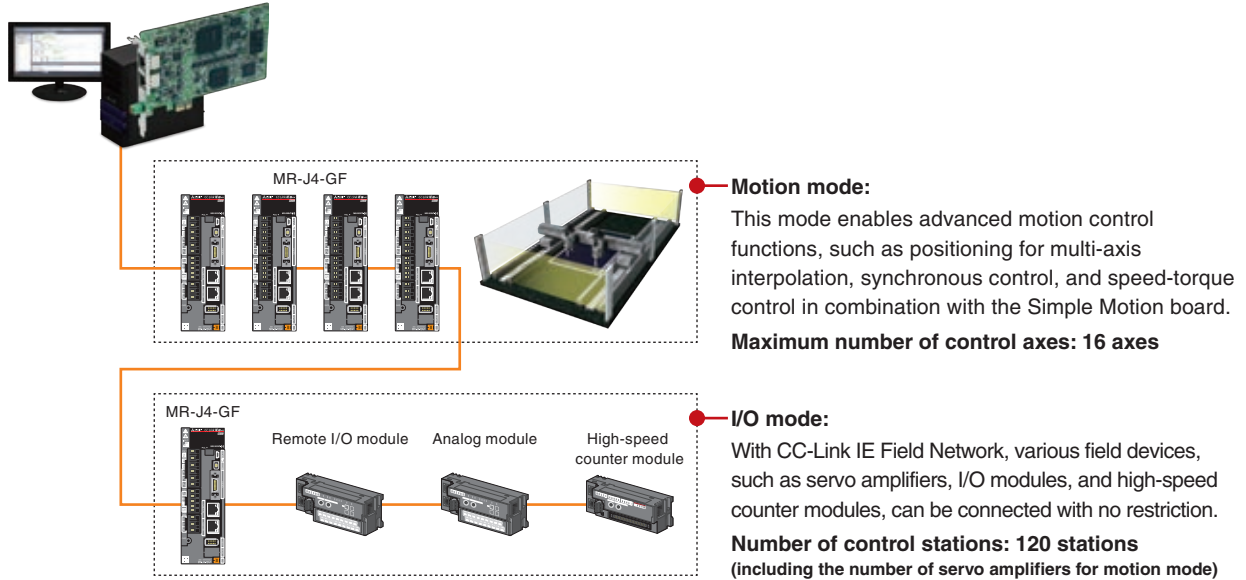
- Various field devices, such as servo amplifiers, I/O modules, and high-speed counter modules are connected flexibly with the same network.
- The Simple Motion board functions as a master station of CC-Link IE Field Network.
- The interrupt function via PCI Express® enables an event-driven program to be created with Visual C++®.



CC-Link IE

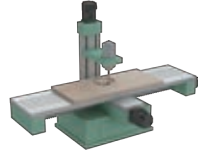
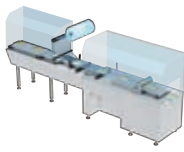
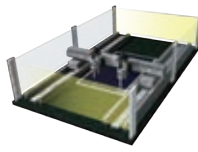

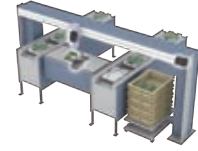

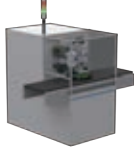
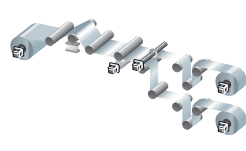
Servo System Configuration

The Simple Motion board is equipped with not only functions for Motion control, but also a function as a master station of CC-Link IE Field Network. Up to 120 stations including servo amplifiers are connectable.



Application Examples

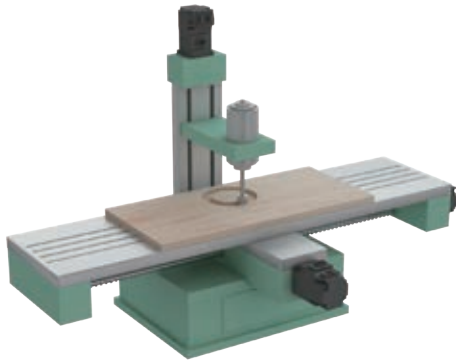
Selecting the best suitable control methods and functions for your machine achieves an optimal solution.

<p>Woodworking machine</p>  <p>Positioning Circular Interpolation Helical Interpolation</p>	<p>Packaging machine</p>  <p>Advanced Sync. Mark Detection</p>	<p>Liquid crystal apparatus</p>  <p>Advanced Sync.</p>	<p>Cutting machine</p>  <p>Cam Auto-Generation Advanced Sync.</p>
<p>Inspection equipment</p>  <p>Shutter Output Positioning</p>	<p>Flip chip bonder</p>  <p>Positioning</p>	<p>Electronic component assembling machine</p>  <p>Positioning</p>	<p>Converting machine</p>  <p>Speed-Torque</p>

Simple Motion Board

Positioning Control

- To respond to various application needs, the Simple Motion board offers various control functions, such as linear interpolation, 2-axis circular interpolation, fixed-pitch feed, and continuous trajectory control.
- Automatic operation can be executed easily by setting positioning addresses, speeds, and other setting items with the API library.
- Powerful sub-functions, such as M-code output, skip, speed change, and target position change, are available.

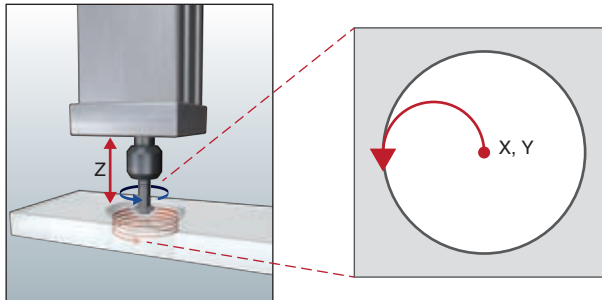


```
void StartPositioningSample( MMC_Axis* axis1 )
{
    unsigned long retCode;

    /* Starts positioning by positioning data No.1 */
    retCode = axis1->StartPositioning( 1 );
    if( retCode != MMC_OK ) { /* Error processing */ }

    /* Waits until completion of positioning control */
    retCode = axis1->WaitPositioningDone
    ( MMC_POSITIONING_DONE_INP, 10000 );
    if( retCode != MMC_OK ) { /* Error processing */ }
}
```

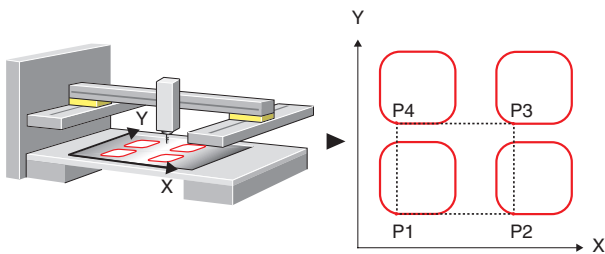
Helical interpolation



For applications that require the boring of deep, large holes, usually multiple interpolation control of three axes (X, Y, and Z) or more is performed.

- The actual milling is done in a circle, with the X and Y axes synchronized to achieve the pre-set size.
- The depth of the hole is simultaneously controlled along the Z axis, ensuring minimal deviation in the cutting bit position.

Block-start



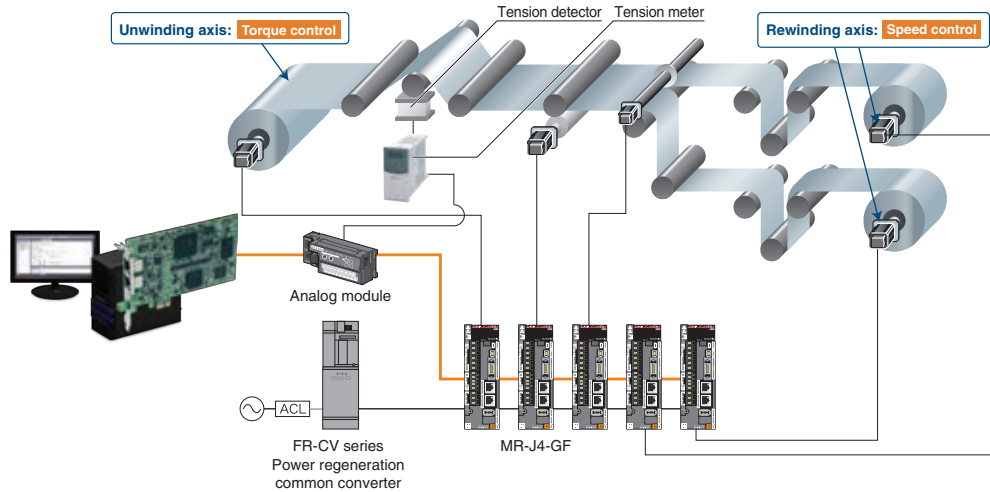
Just with a one-time start, the operation is carried out sequentially following the multiple positioning data.

This control is suitable for machines requiring the same operation repeated.

Positioning operation starts from Block start data No.1, and four squircles are drawn in this example.

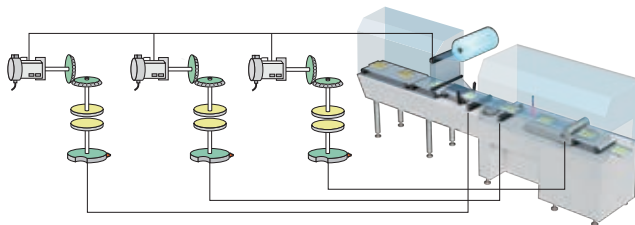
Speed-Torque Control

The Simple Motion board can be used for the speed-torque control, such as unwinding or rewinding. Positioning using absolute position coordinates can be smoothly performed even after switching back to position control because the current position is controlled during the speed-torque control.



Advanced Synchronous Control

The advanced synchronous control is software-based synchronous control as an alternative to mechanical control, such as gear, shaft, clutch, speed change gear, and cam. In addition, a cam is easily generated with cam auto-generation function. The synchronous control can be started and ended for each axis, allowing the synchronous control axis and positioning control axis within the same program.



```

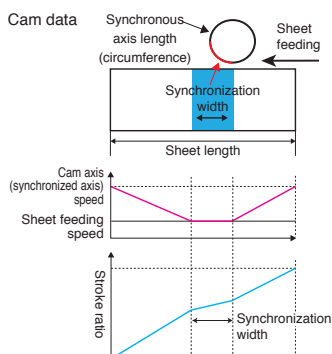
void SynchronizationSample
( MMC_Controller *controller,
  MMC_Axis *axis1, MMC_Axis *axis4 )
{
  unsigned long retCode;
  /* Starts synchronous control */
  retCode = axis1->StartSync();
  if( retCode != MMC_OK ) { /* Error processing */ }

  /* Starts JOG operation of virtual servo axis */
  retCode = axis4->StartJog( 20000 );
  if( retCode != MMC_OK ) { /* Error processing */ }

  :

  /* Stops synchronous control */
  axis1->StopSync( );
  if( retCode != MMC_OK ) { /* Error processing */ }
}
    
```

Cam auto-generation



Cam data for a rotary cutter can be generated automatically simply by parameter settings.

Software Development Kit MELSOFT EM Software Development Kit

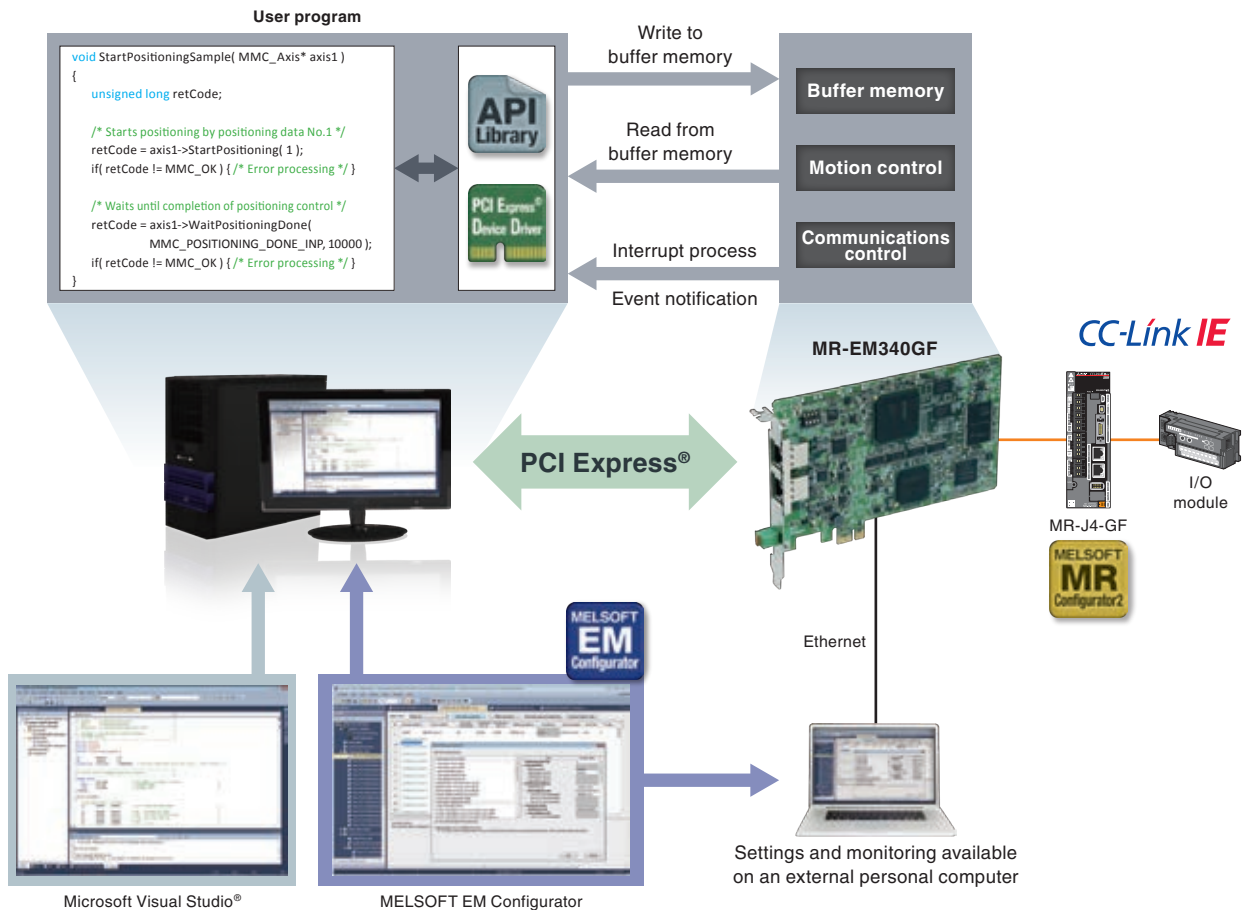


MELSOFT EM Software Development Kit is a development software package, supporting the engineering process from system design and programming to debug and maintenance for the Simple Motion board.

- [Included software]
- MELSOFT EM Configurator
 - MELSOFT MR Configurator2
 - API library
 - PCI Express® device driver

Development and Debugging Environments

A user program is created by adding the API library (for motion control) to a project of Microsoft Visual Studio®.



(Note): OS and the development environment are not included.



MELSOFT EM Configurator

Every step in the engineering process from system design and programming to debug and maintenance, is supported by this software.



MELSOFT MR Configurator2

Primarily, tuning, monitoring, and diagnosis are easily performed with this software by being connected to a servo amplifier.



API library

The API library is an add-on library which uses functions (method) and labels (member) of controller and axis classes, and enables easy programming with Visual C++®.

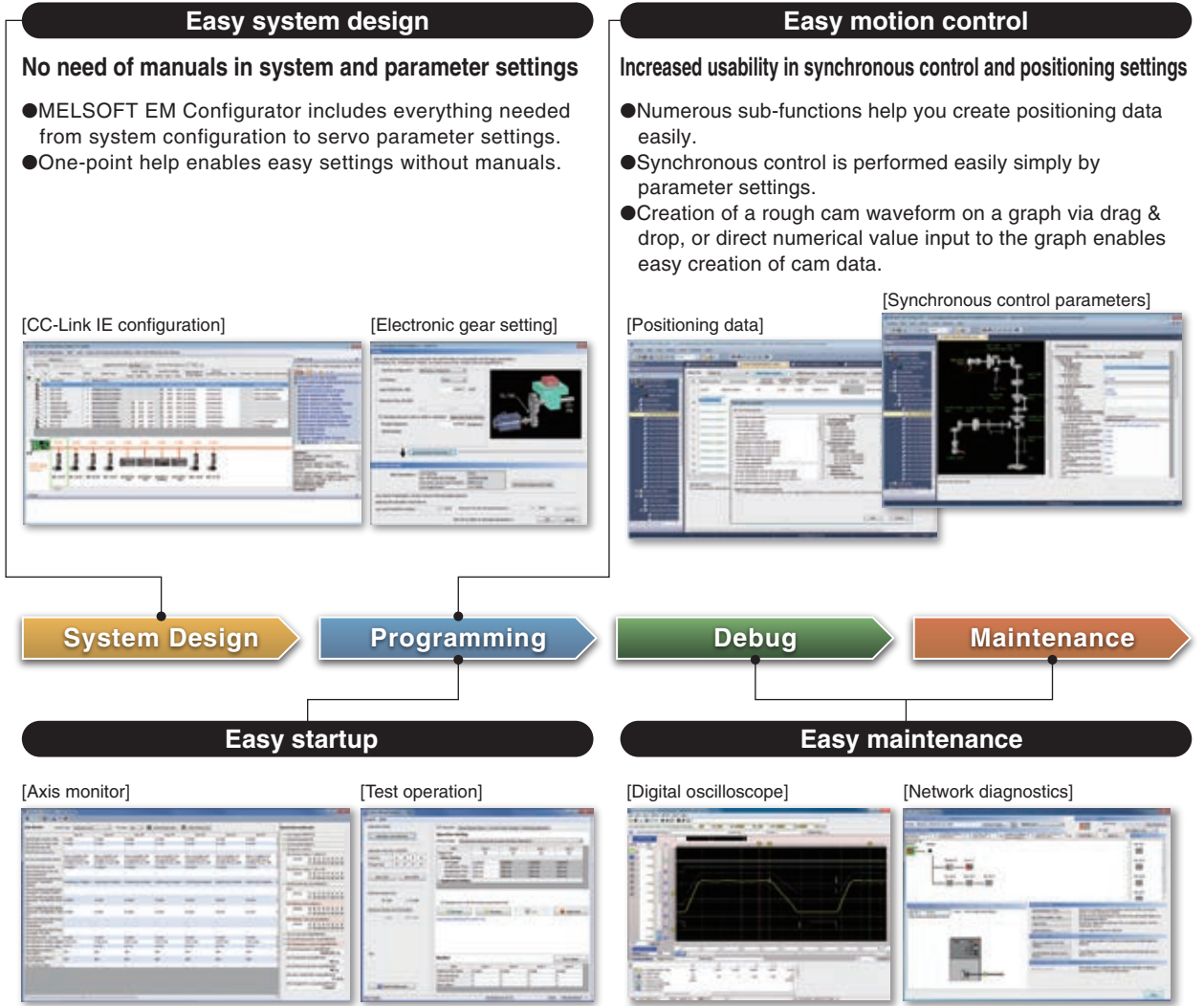


PCI Express® device driver

The PCI Express® device driver is software for a user program to gain access to the Simple Motion board via PCI Express®.



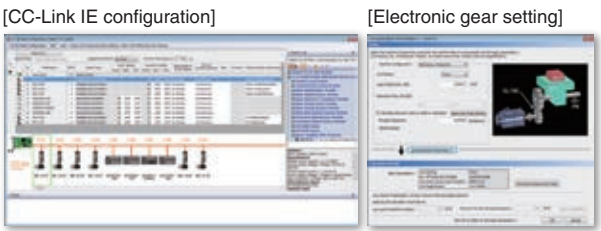
MELSOFT EM Configurator (Setting Tool for Simple Motion Board)



Easy system design

No need of manuals in system and parameter settings

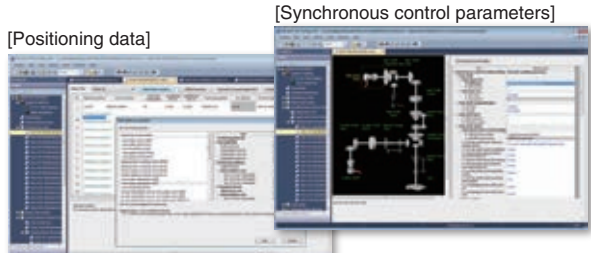
- MELSOFT EM Configurator includes everything needed from system configuration to servo parameter settings.
- One-point help enables easy settings without manuals.



Easy motion control

Increased usability in synchronous control and positioning settings

- Numerous sub-functions help you create positioning data easily.
- Synchronous control is performed easily simply by parameter settings.
- Creation of a rough cam waveform on a graph via drag & drop, or direct numerical value input to the graph enables easy creation of cam data.



System Design

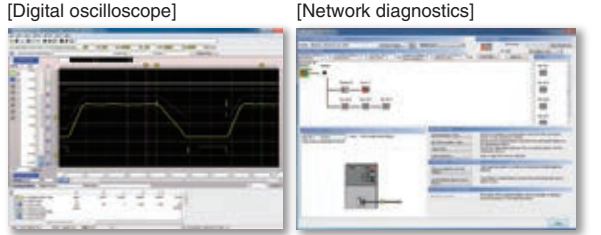
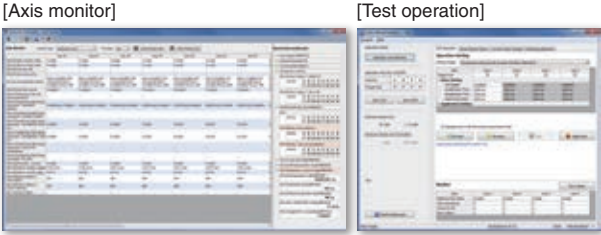
Programming

Debug

Maintenance

Easy startup

Easy maintenance



Increased efficiency in debugging and maintenance

- A customizable axis monitor increases efficiency in startup.
- An operation check of servo motors is possible by test operation before creating a program.

A wide variety of diagnosis functions

- Waveform display on a digital oscilloscope supports troubleshooting.
- Network errors are displayed with Network diagnostics.

Available in the future

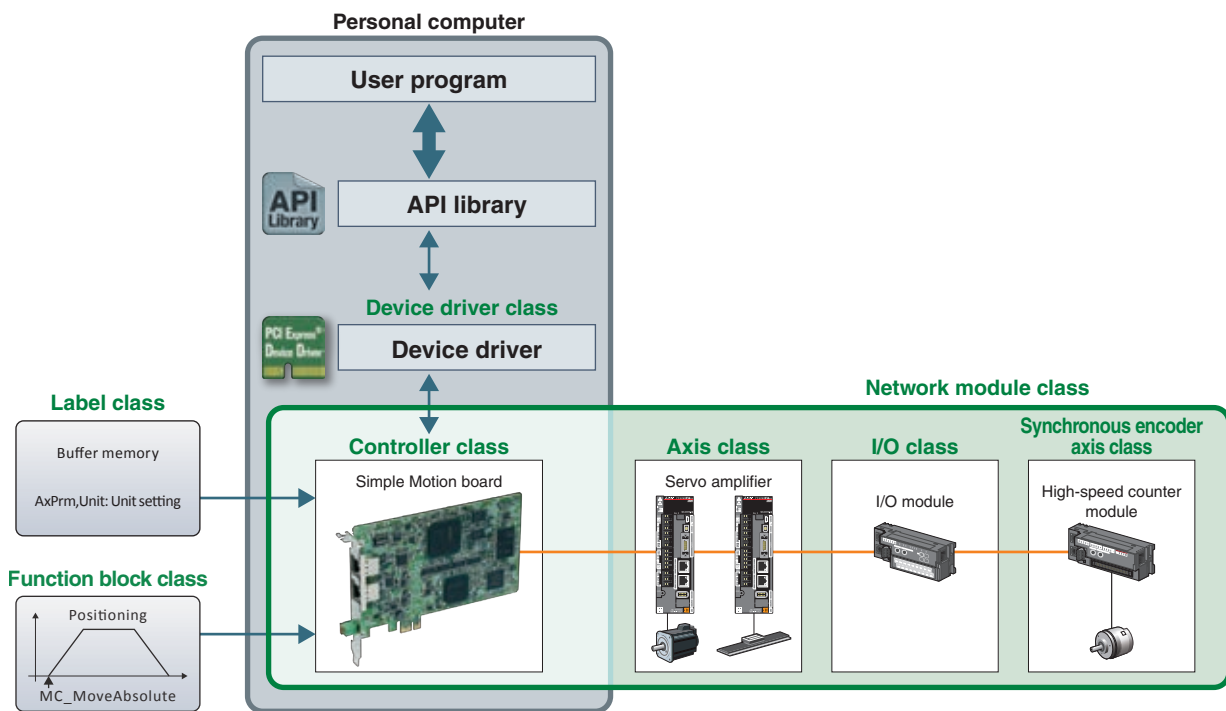


API library (C++ motion class library)

With the C++ motion class library, a program is created using functions (method) and labels (member) of controller and axis classes, and other classes.

- The class library creates the program with high readability.
- Coding time is reduced with Microsoft Visual Studio® IntelliSense®.
- Compatibility with event-driven programming is improved by specifying any bit data as a condition of interruption.
- The library with the same interface specifications as PLCopen® Motion Control FB, is available and suitable for fixed-cycle programming.

[Conception diagram of class types in C++ motion class library]



Programming using motion functions and axis labels

```
void ObjectSample( MMC_DeviceDriver *pciDev,
MMC_Controller* controller, MMC_Axis* axis1 )
{
    long data;
    unsigned long retCode;
    /* Generates PCI Express® device driver class objects */
    retCode = MmfCreatePciDevice( boardID, &pciDev );

    /* Generates MMC_EM340GF class objects */
    retCode = MmfCreateEM340GF( pciDev,
(MMC_EM340GF **)&controller );
    /* Gets axis class objects */
    retCode = controller->GetAxis( 1, &axis1 );

    /* Sets parameters for acceleration time constant=0 */
    axis1->AxPrm.AccelerationTime0 = 1000;
    /* Gets the actual current value */
    data = axis1->AxMntr.ActualPosition;

    /* Starts positioning by positioning data No.1 */
    retCode = axis1->StartPositioning( 1 );
    if( retCode != MMC_OK ) { /* Error processing */ }
}
```

Microsoft Visual Studio® IntelliSense®

```
void Sample( MMC_Axis* axis1 )
{
    axis1->AxMntr.
}
```

The complete word (IntelliSense) function lists the axis class motion functions (method) and axis labels (member) that can be used.

Event-driven programming (CPU resources are maximally used)

```

void InterruptSample( MMC_Axis* axis1 )
{
    unsigned long retCode;
    MMST_PositioningData positioningData = { 0 };

    /* Structures positioning data No.1 */
    positioningData.OperationPattern = 0;           /* [Da.1] Operation pattern */
    positioningData.ControlMethod = 0x01;          /* [Da.2] Control method */
    positioningData.AccelerationTimeNo = 0;        /* [Da.3] Acceleration time No. */
    positioningData.DecelerationTimeNo = 0;        /* [Da.4] Deceleration time No. */
    positioningData.PositioningAddress = -1000000; /* [Da.6] Positioning address */
    positioningData.CommandSpeed = 20000;          /* [Da.8] Command speed */

    /* Sets positioning data No.1 */
    axis1->SetPositioningData( 1, positioningData );

    /* Sets the interrupt event of positioning completion to a nonsignaled state */
    retCode = axis1->ResetPositioningDoneIntEvent( MMC_POSITIONING_DONE_INP );
    if( retCode != MMC_OK ) { /* Error processing */ }

    /* Starts positioning by positioning data No.1 */
    retCode = axis1->StartPositioning( 1 );
    if( retCode != MMC_OK ) { /* Error processing */ }

    /* Waits until completion of positioning control */
    retCode = axis1->WaitPositioningDoneIntEvent( MMC_POSITIONING_DONE_INP, 10000 );
    if( retCode != MMC_OK ) { /* Error processing */ }
}

```

Waits until the positioning complete interrupt event is in a signaled state.

Fixed-cycle programming (API library with the same interface specifications as PLCopen® Motion Control FB)

The library is effective when the ST language is replaced with the C language or when the program cannot be in a wait state inside methods to keep the constant scan time.

```

void FunctionBlockSample( MC_MoveAbsolute *fbMC_MoveAbsolute, AXIS_REF *axis, int phase )
{
    switch( phase )
    {
        case 0:

            /* Executes MC_MoveAbsolute */
            fbMC_MoveAbsolute->Axis = axis;           /* Axis information */
            fbMC_MoveAbsolute->PositionDataNo = 1;    /* Positioning data No. */
            fbMC_MoveAbsolute->Position = -1000.0;    /* Target position */
            fbMC_MoveAbsolute->Velocity = 20.00;      /* Speed */
            fbMC_MoveAbsolute->Acceleration = 1000;   /* Acceleration time */
            fbMC_MoveAbsolute->Deceleration = 1000;   /* Deceleration time */
            fbMC_MoveAbsolute->Direction = 1;         /* Rotation direction */
            fbMC_MoveAbsolute->Execute = true;        /* Execute command ON */
            fbMC_MoveAbsolute->Update();              /* Executes FB */
            if( fbMC_MoveAbsolute->Error ) { /* Error processing */ }
            if( fbMC_MoveAbsolute->Done )
            {
                phase = 1;
            }
            break;

        case 1:
            fbMC_MoveAbsolute->Execute = false;      /* Execute command OFF */
            fbMC_MoveAbsolute->Update();              /* Executes FB */
            phase = 2;
            break;

        case 2:
            :
            :
            break;
    }
}

```

Programming to start positioning

Positioning is started simply by setting positioning data to the Simple Motion board with the API library.

```

void InterruptSample( MMC_Axis* axis1 )
{
    unsigned long retCode;
    MMST_PositioningData positioningData = { 0 };

    /* Structures positioning data No.1 */
    positioningData.OperationPattern = 0;           /* [Da.1] Operation pattern */
    positioningData.ControlMethod = 0x01;         /* [Da.2] Control method */
    positioningData.AccelerationTimeNo = 0;       /* [Da.3] Acceleration time No. */
    positioningData.DecelerationTimeNo = 0;      /* [Da.4] Deceleration time No. */
    positioningData.PositioningAddress = -1000000; /* [Da.6] Positioning address */
    positioningData.CommandSpeed = 20000;        /* [Da.8] Command speed */

    /* Sets positioning data No.1 */
    axis1->SetPositioningData( 1, positioningData );

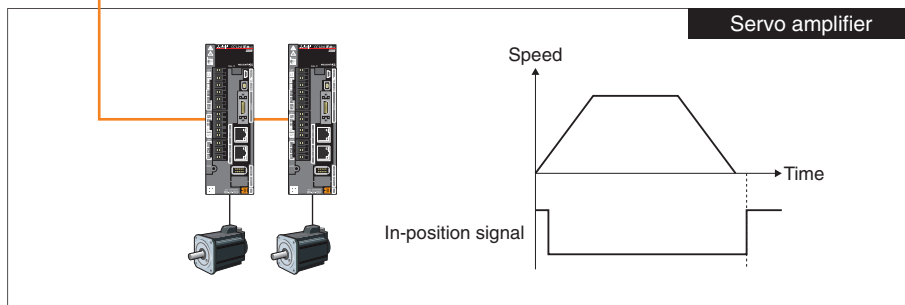
    /* Sets the interrupt event of positioning completion to a nonsignaled state. */
    retCode = axis1->ResetPositioningDoneIntEvent( MMC_POSITIONING_DONE_INP );
    if( retCode != MMC_OK ) { /* Error processing */ }

    /* Starts positioning by positioning data No.1 */
    retCode = axis1->StartPositioning( 1 );
    if( retCode != MMC_OK ) { /* Error processing */ }

    /* Waits until completion of positioning control */
    retCode = axis1->WaitPositioningDoneIntEvent( MMC_POSITIONING_DONE_INP, 10000 );
    if( retCode != MMC_OK ) { /* Error processing */ }
}
    
```

Simple Motion board

Axis 1		Axis 2					
No.	[Da.1] Operation pattern	[Da.2] Control method	[Da.3] Acceleration time No.	[Da.4] Deceleration time No.	[Da.5] Positioning address	[Da.8] Command speed	...
1	0	0x01	0:1000	0:1000	-1000000	20000	0
2	0	0x08	0:1000	0:1000	200000	10000	0



Main API library list

MMC_Controller Class

Get object method	
GetAxis	Gets the object of the axis class.
GetSlavelo	Gets the object of the I/O class.
GetSyncEncoder	Gets the object of the synchronous encoder axis class.
System method	
ResetController	Executes remote RESET.
SetUserProgramReady	Sets the user program ready signal [Y0].
Interrupt method	
SetInterruptParameter	Sets the interrupt parameter.
EnableInterrupt	Enables the interrupt output.
DisableInterrupt	Disables the interrupt output.
Synchronous control method	
CalcCamCommandPosition	Calculates cam axis feed current value.
CalcCamCommandPositionPerCycle	Calculates cam axis current value per cycle.
MakeRotaryCutterCam	Auto-generates the cam (central reference) for rotary cutter.
MakeEasyStrokeRatioCam	Auto-generates the easy stroke ratio cam.
MakeAdvancedStrokeRatioCam	Auto-generates the advanced stroke ratio cam.

MMC_Axis Class

Positioning data method	
SetPositioningData	Sets the positioning data.
SetBlockStartData	Sets the block start data.
SetBlockConditionData	Sets the condition data used by block start.
GetPositioningData	Gets the positioning data.
GetBlockStartData	Gets the block start data.
GetBlockConditionData	Gets the condition data used by block start.
Operation method	
StartPositioning	Starts positioning control.
StartBlockPositioning	Starts advanced positioning control.
StopPositioning	Stops axis.
RestartPositioning	Restarts stopped axis.
WaitPositioningDone	Waits until completion of positioning control.
ResetPositioningDoneIntEvent	Sets the positioning complete interrupt event to a nonsignaled state.
SetPositioningDoneIntEvent	Sets the positioning complete interrupt event to a signaled state.
WaitPositioningDoneIntEvent	Waits until the positioning complete interrupt event is in a signaled state.
StartJog	Starts JOG operation.
StopJog	Stops JOG operation.
EnableMPG	Enables manual pulse generator operation.
DisableMPG	Disables manual pulse generator operation.
ChangeControlMode	Changes control mode.
Change method	
ChangeSpeed	Changes speed and acceleration/deceleration time.
ChangePosition	Changes target position and command speed.
Interrupt method	
SetInterruptParameter	Sets the interrupt parameter.
ResetIntEvent	Sets the interrupt event to a nonsignaled state.
SetIntEvent	Sets the interrupt event to a signaled state.
WaitIntEvent	Waits until the interrupt event is in a signaled state.
Synchronous control method	
StartSync	Starts synchronous control.
StopSync	Stops synchronous control.
ChangeSyncPosition	Changes current value during synchronous control.
MoveCamPosition	Moves cam axis during synchronous control.

MC_FunctionBlock Class

MC_Power	Changes the servo amplifier of the specified axis to an operable state.
MCv_Home	Executes home position return.
MC_Stop	Stops the specified axis.
MC_MoveAbsolute	Specifies the absolute target position of the specified axis and executes positioning.
MC_MoveRelative	Moves the specified distance from the current position.
MC_Reset	Cancel the errors and warnings of the specified axis.
MC_MoveAdditive	Adds the most recent relative position specified by the positioning command of the specified axis, and executes positioning.
MC_MoveVelocity	Executes speed control for the specified axis at the specified speed.
MC_TorqueControl	Executes torque control for the specified axis at the specified torque.
MC_SetPosition	Changes the current position (command position and feedback position) of the specified axis.
MC_SetOverride	Changes the target speed of the specified axis.



CC-Link IE Field Network
Servo Amplifier

MR-J4-GF

MR-J4-GF-RJ

CC-Link IE Field Network servo amplifiers achieve an optimal solution and improve productivity in combination with the Simple Motion board.

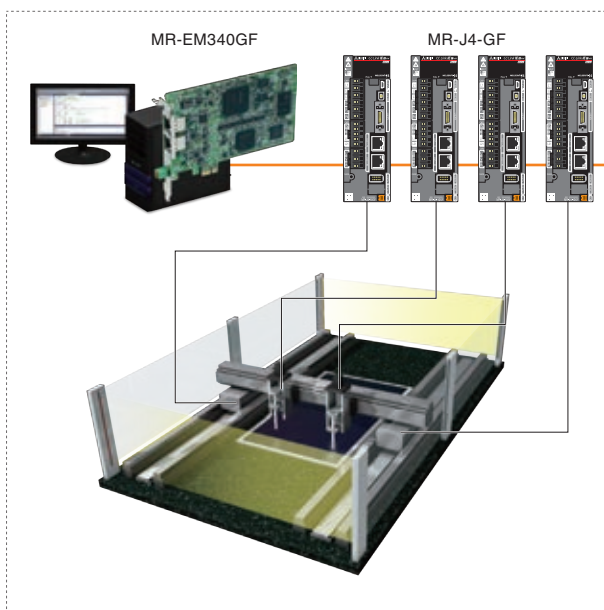
- **Industry-leading basic performance**
Industry-leading levels of servo amplifier basic performance shorten a machine cycle time.
- **Advanced servo gain adjustment**
The advanced vibration suppression control function is easily used for maximizing your machine performance.
- **A wide range of product series and capacities**
From rotary to linear and direct drive motors, a wide range of servo motors are available, significantly improving your machine performance.
- **Preventive maintenance**
The data inside a servo amplifier are read via the network, and used for preventive maintenance, such as machine diagnostics.

Control mode

Two types of modes are available according to your needs:

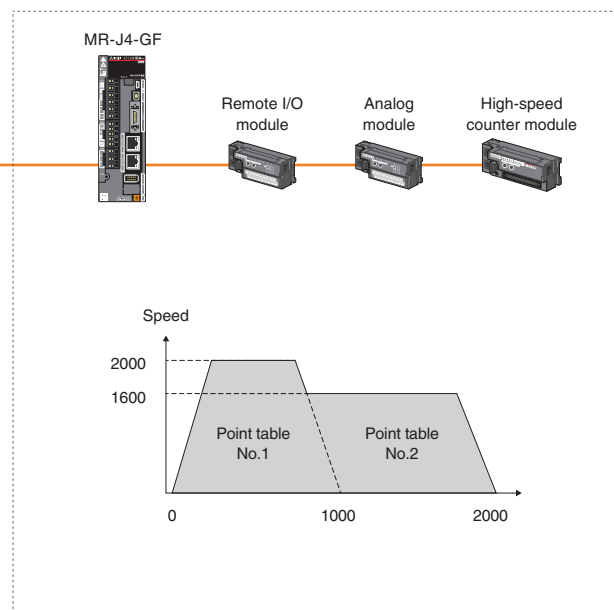
- Motion mode for a wide range of motion control functions, such as multiple-axis positioning, synchronous control, etc.
- I/O mode for single-axis positioning

Motion mode



This mode enables advanced motion control functions, such as multi-axis positioning, synchronous control, and speed-torque control.

I/O mode

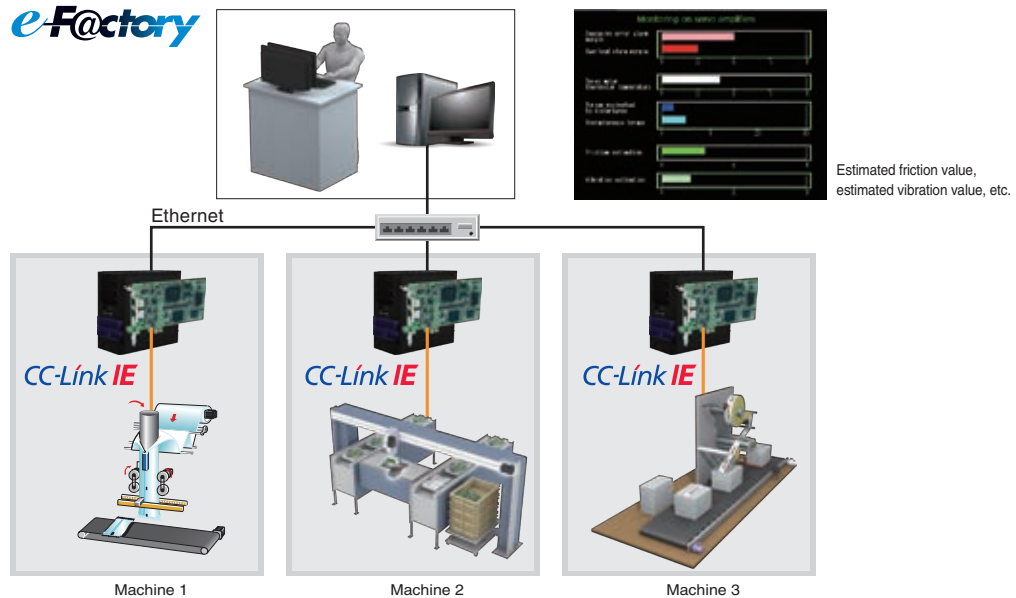


This mode easily drives a belt conveyor, a rotary table, a ball screw mechanism, etc. by using the built-in positioning function in a servo amplifier.

Direct Access to IT System

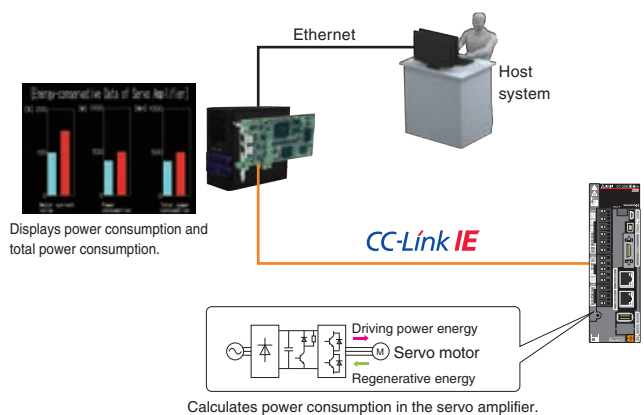
Data of servo amplifiers and servo motors for each machine can be collected via CC-Link IE Field Network. The status of the entire product line can be visualized by batch management of the collected data. A CC-Link IE Field Network servo system supports to build IoT ^(Note-1) for your machine.

(Note-1): IoT (Internet of Things)



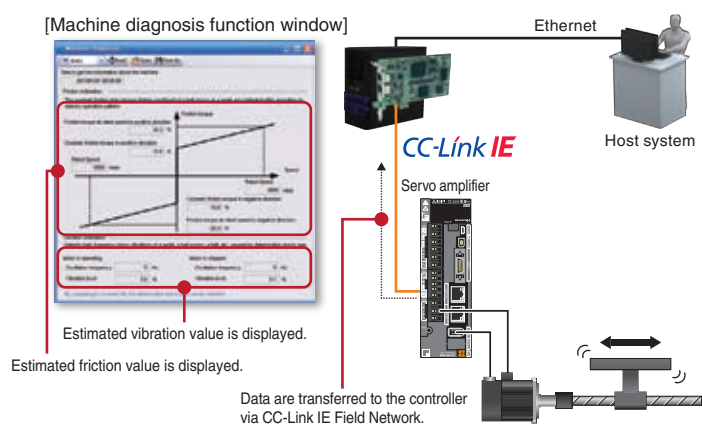
Monitoring of Servo Data

Servo data up to fifty monitoring items can be monitored and modified successively during operation. The operation status of servo amplifiers and servo motors acquired via CC-Link IE Field Network is transferred and displayed on the host system.

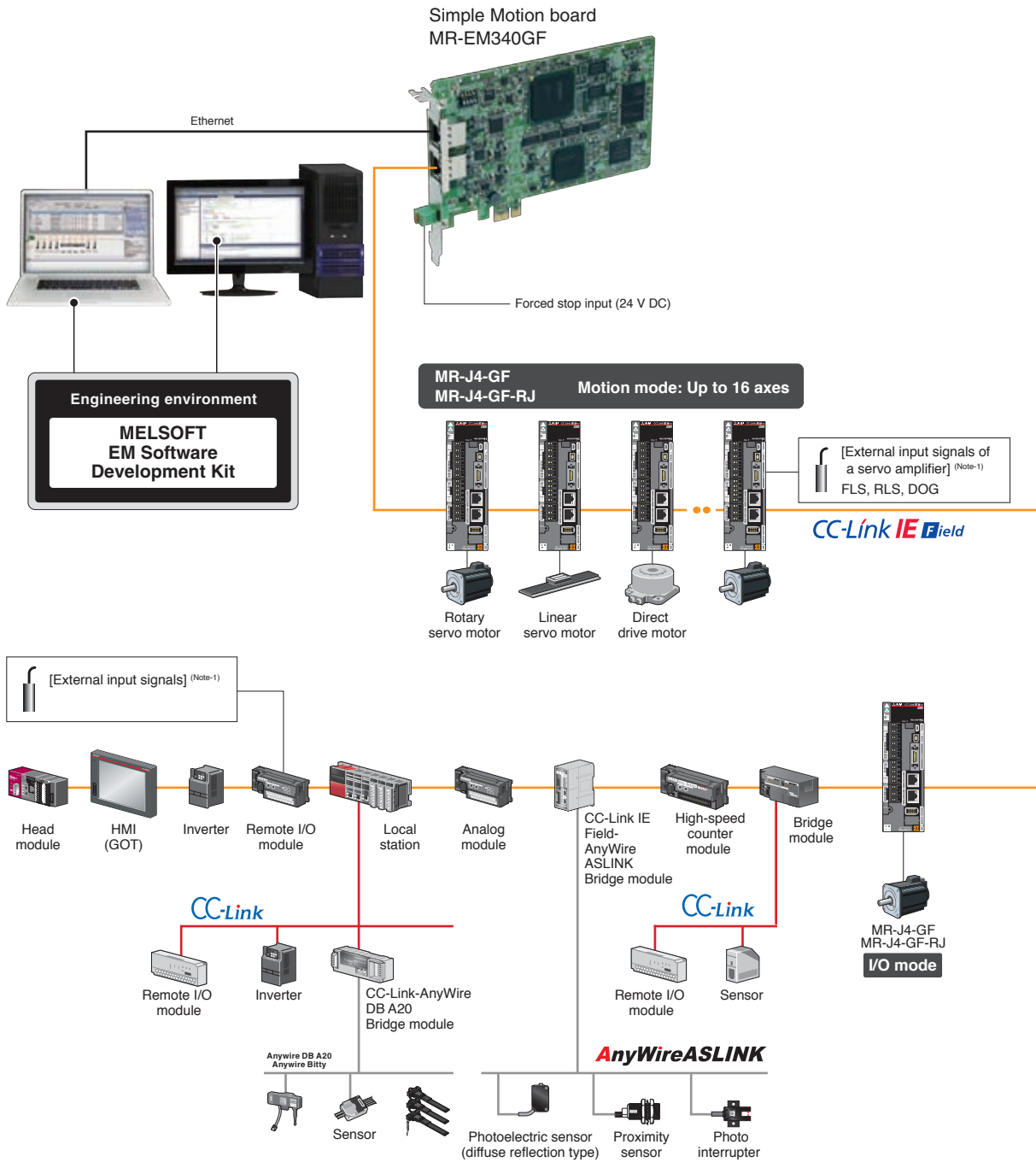


Preventive Maintenance

Machine diagnosis function detects changes in mechanical components (ball screw, guide, bearing, belt, etc.) by analyzing machine friction, load moment of inertia, unbalanced torque, and changes in vibration components using the data inside a servo amplifier, supporting timely maintenance of these components. In addition, the data are transferred to a host system and used to monitor the entire line.



System configuration



(Note-1): An input destination of external input signals (FLS, RLS, and DOG) is changed by parameters.

Slave station: 120 stations (including the number of motion mode compatible servo amplifiers)

(Note): A switching hub is required for star topology.

Control specifications

Item		Specification	
		MR-EM340GF	
Maximum number of control axes (virtual servo amplifier axis included)		16 axes	
Operation cycle (operation cycle settings)		0.5 ms, 1.0 ms, 2.0 ms, 4.0 ms	
Interpolation function		Linear interpolation (up to 4 axes), Circular interpolation (2 axes), Helical interpolation (3 axes)	
Control modes		Positioning, Trajectory control (linear, arc, and helical), Speed control, Speed-torque control	
Acceleration/deceleration process		Trapezoidal acceleration/deceleration, S-curve acceleration/deceleration	
Compensation function		Backlash compensation, Electronic gear, Near pass function	
Synchronous control		Synchronous encoder input, Cam, Phase compensation, Cam auto-generation	
Control unit		mm, inch, degree, pulse	
Number of positioning data		600 data/axis	
Backup		Parameters, positioning data, and block start data can be saved on flash ROM (battery-less backup)	
Home position return	Home position return method	Driver home position return method	
	Fast home position return control	Provided	
	Sub-function	Provided (the sub-function of a servo amplifier)	
Positioning control	Linear control	Linear interpolation control (up to 4 axes) ^(Note-1) (vector speed, reference axis speed)	
	Fixed-pitch feed	Fixed-pitch feed control	
	2-axis circular interpolation	Auxiliary point-specified circular interpolation, Central point-specified circular interpolation	
	Speed control	Speed control	
	Speed-position switching	INC mode, ABS mode	
	Position-speed switching	INC mode	
	Current value change	Positioning data, Start No. for a current value changing	
	NOP instruction	Provided	
	JUMP instruction	Conditional JUMP, Unconditional JUMP	
	LOOP, LEND	Provided	
High-level positioning		Block start, Condition start, Wait start, Simultaneous start, Repeated start	
Manual control	JOG operation	Provided	
	Inching operation	Provided	
	Manual pulse generator	Possible to connect 1 module (incremental), Unit magnification (1 to 10000 times) Via link device	
Expansion control	Speed-torque control	Speed control without positioning loops, Torque control	
Absolute position system		Made compatible by setting a battery to a servo amplifier	
Synchronous encoder interface		16CH	
	Via buffer memory	Provided (incremental)	
	Link device	Provided (incremental)	
	Via servo amplifier	16CH	
Functions that limit control	Speed limit	Speed limit value, JOG speed limit value	
	Torque limit	Torque limit value same setting, torque limit value individual setting	
	Forced stop	Internal interface	Provided
		Buffer memory	Provided
		Link device	Provided
	Software stroke limit	Movable range check with current feed value, movable range check with machine feed value	
Hardware stroke limit	Provided		
Functions that change control details	Speed change	Provided	
	Override	0 to 300 [%]	
	Acceleration/deceleration time change	Provided	
	Torque change	Provided	
	Target position change	Target position address and speed are changeable	
Other functions	M-code output	WITH mode/AFTER mode	
	Step function	Deceleration unit step, Data No. unit step	
	Skip function	Via buffer memory, Via external command signal	
	Teaching function	Provided	
Parameter initialization function		Provided	
External input signal setting function	Via buffer memory	Provided	
	Link device	Provided	
	Via servo amplifier	Provided	
Amplifier-less operation function (virtual servo amplifier function)		Provided	
Mark detection function	Continuous Detection mode, Specified Number of Detections mode, Ring Buffer mode		
	Mark detection signal	Up to 16 points ^(Note-3)	
Digital oscilloscope function ^(Note-2)	Mark detection setting	16 settings	
	Bit data	16CH	
Word data		16CH	

(Note-1): 4-axis linear interpolation control is enabled only at the reference axis speed.

(Note-2): 8CH word data and 8CH bit data are displayed in real time.

(Note-3): The Mitsubishi Electric remote input module is required.

Specifications

Simple Motion board specifications

Item		Specification
		MR-EM340GF
Servo amplifier connection system		CC-Link IE Field Network
Maximum distance between stations [m(ft.)]		100 (328.08)
Peripheral I/F		Ethernet (100BASE)
Forced stop input signal (EM)	Number of input points	1 point
	Input method	Positive Common/ Negative Common Shared Type (Photocoupler isolation)
	Rated input voltage/current	24 V DC/approx. 2.4 mA
	Operating voltage range	20.4 to 26.4 V DC (24 V DC +10 %/-15 %, ripple ratio 5 % or less)
	ON voltage/current	17.5 V DC or more/2.0 mA or more
	OFF voltage/current	1.8 V DC or less/0.18 mA or less
	Input resistance	Approx. 10 kΩ
	Response time	1 ms or less (OFF to ON, ON to OFF)
Recommended wire size [mm ²]		0.08 to 0.5 (AWG 20 to AWG 28)
Number of Simple Motion boards for one computer		4
Bus specification		PCI Express® 2.0 × 1
	Size [mm(inch)]	Short sized version (167.65(6.60) × 111.15(4.38))
Power supply voltage		12 V DC/3.3 V DC
Current consumption [A]	12 V DC	0.4
	3.3 V DC	0.6
Mass [kg]		0.13

Operation environment for MELSOFT EM Development Kit

Item	Description	
Personal computer	Personal computer	Microsoft® Windows® supported personal computer
	OS	Microsoft® Windows® 8.1 (Pro, Enterprise) English version (64-bit/32-bit) Microsoft® Windows® 7 (Professional, Ultimate, Enterprise) English version (64-bit/32-bit) [Service Pack 1]
	CPU	Desktop: Intel® Celeron® Processor 2.8 GHz or more recommended Laptop: Intel® Pentium® M Processor 1.7 GHz or more recommended
	Required memory	1 GB or more recommended (For 32-bit edition) 2 GB or more recommended (For 64-bit edition)
Available hard disk space	When installing the test tool: 3 GB or more of available hard disk space required When operating the test tool: 512 MB or more of available hard disk space required	
Disk drive	DVD-ROM supported disk drive	
Monitor	Resolution 1024 × 768 pixels or higher	
Communications interface	PCI Express® BUS Ethernet port	

Development environment

Item	Description
OS for user program operation	The same operation environment as MELSOFT EM Software Development Kit
Software development environment	Microsoft® Visual C++® 2013/2012/2010
API library	Class library (Only compiled into C++)

Performance specifications of CC-Link IE Field Network

Item		Specification		
		MR-EM340GF		
Maximum link points per network	RX	16k points (16384 points, 2 kbytes)		
	RY	16k points (16384 points, 2 kbytes)		
	RWr	8k points (8192 points, 16 kbytes)		
	RWw	8k points (8192 points, 16 kbytes)		
Maximum link points per station	Master station	RX	16k points (16384 points, 2 kbytes)	
		RY	16k points (16384 points, 2 kbytes)	
		RWr	8k points (8192 points, 16 kbytes)	
		RWw	8k points (8192 points, 16 kbytes)	
	Local station	RX	2k points (2048 points, 256 bytes)	
		RY	2k points (2048 points, 256 bytes)	
		RWr	256 points, 512 bytes	
		RWw	256 points, 512 bytes	
	Intelligent device station	RX	2k points (2048 points, 256 bytes)	
		RY	2k points (2048 points, 256 bytes)	
		RWr	256 points, 512 bytes	
		RWw	256 points, 512 bytes	
Remote device station	RX	128 points, 16 bytes		
	RY	128 points, 16 bytes		
	RWr	64 points, 128 bytes		
	RWw	64 points, 128 bytes		
Ethernet	Communication speed		1 Gbps	
	Connection cable		1000BASE-T Ethernet cable ^(Note-1) ; category 5e or higher (double shielded/STP) straight cable	
	Maximum distance between stations [m(ft.)]		100(328.08) (conforms to ANSI/TIA/EIA-568-B (category 5e))	
	Topology		Line type, star type, line/star mixed type	
Overall cable distance	Line type [m(ft.)]		12000(39370.08) (When 1 master station and 120 slave stations are connected)	
	Star type ^(Note-2)		Depends on system configuration	
Maximum connectable stations per network		121 stations (1 master station, 120 slave stations)		
Maximum number of networks		239		

(Note-1): Use the cables recommended by CC-Link Partner Association for CC-Link IE Field Network.

CC-Link IE Controller Network cables are not compatible with CC-Link IE Field Network.

(Note-2): A switching hub is required for star topology.

Ethernet Cable Specifications

Item		Specification
Ethernet cable	Category 5e or higher (double shielded/STP) straight cable	
	Standard	The cable must meet the following standards: <ul style="list-style-type: none"> • IEEE802.3 (1000BASE-T) • ANSI/TIA/EIA-568-B (category 5e)
	Connector	RJ-45 connector with shield

Products on the Market

Ethernet Cable

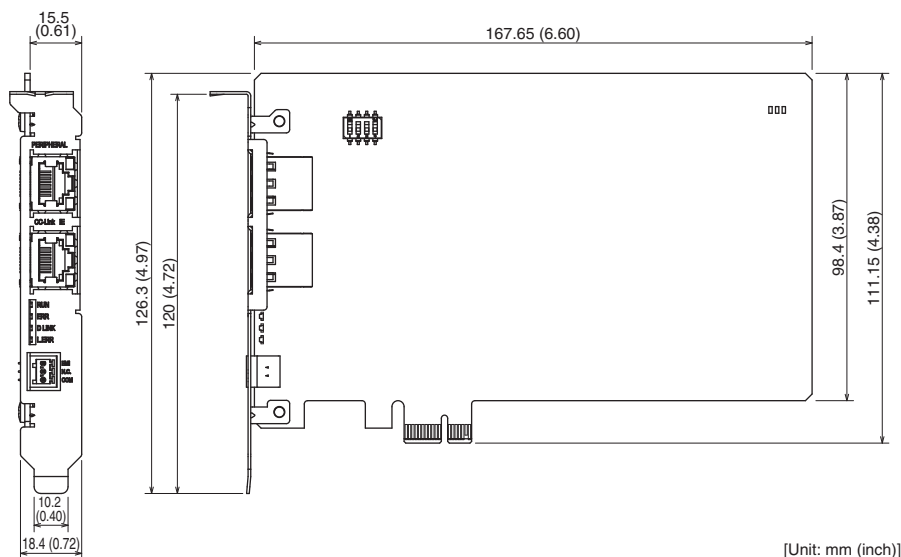
Item	Model	Specification	Note
Ethernet cable	For indoor	SC-E5EW-S_M	_: cable length (100 m max., unit of 1 m)
	For indoor and moving part	SC-E5EW-S_M-MV	_: cable length (45 m max., unit of 1 m)
	For indoor/outdoor	SC-E5EW-S_M-L	_: cable length (100 m max., unit of 1 m)

Microsoft, Windows, Visual C++, Visual Studio, and IntelliSense are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
 Intel, Pentium, and Celeron are registered trademarks of Intel Corporation in the U.S. and/or other countries.
 PCI Express® is a registered trademark of PCI-SIG.
 Ethernet is a registered trademark of US Xerox Corporation.
 All other company names and product names used in this document are trademarks or registered trademarks of their respective companies.

Personal Computer Embedded Type Servo System Controller CC-Link IE Simple Motion Board/ MELSOFT EM Software Development Kit

Exterior dimensions

Simple Motion board MR-EM340GF



Product list

Name	Model	Description	Standard
Simple Motion Board	MR-EM340GF	Up to 16 axes	CE, UL, KC
MELSOFT EM Software Development Kit ^(Note-1)	SW1DND-EMSDK-B	<ul style="list-style-type: none"> ● MELSOFT EM Configurator (setting tool) ● MELSOFT MR Configurator2 ● API library (C++ motion class library) ● PCI Express® device driver (including the driver for interrupt) 	

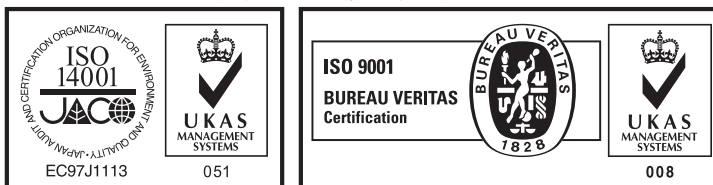
(Note-1): MELSOFT EM Software Development Kit is sold separately.



Safety Warning

To ensure proper use of the products listed in this catalog, please be sure to read the instruction manual prior to use.

Mitsubishi Electric Corporation Nagoya Works is a factory certified for ISO14001 (standards for environmental management systems) and ISO9001 (standards for quality assurance management systems)



MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE: TOKYO BUILDING, 2-7-3, MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS: 1-14, YADA-MINAMI 5, HIGASHI-KU, NAGOYA, JAPAN