Okuma America Corporation

THINC-API
Release Notes For Machining Center

Document No.: S5015-003-20

Revision History

Date	Version	Description	Author
5/09/2007	S5015-003-00	Public release for Machining Center THINC-API Release 1.0.0.0	LHuynh
5/11/2007	S5015-003-01	Revise Public release for Machining Center THINC-API Release 1.0.0.0: - Remove section 3.0 Compatible and put in Installation Manual document - Remove section 4.0 Upgrades and put in THINC-API Help file. - Revise section 6.1 and 6.2 - Use api@okuma.com email instead of p100issues@okuma.com	LHuynh
5/21/2007	S5015-003-02	Revise General defect – Cannot support 0.1 Micron option. Revise 4.1 section	LHuynh
06/04/2007	S5015-003-03	Public release for Machining Center THINC-API Release 1.1.0.0	LHuynh
10/19/2007	S5015-003-04	Public release for Machining Center THINC-API Release 1.3.0.0	LHuynh
2/22/2008	S5015-003-05	Public release for Machining Center THINC-API Release 1.6.0.0	LHuynh
04/11/2008	S5015-003-06	Public release for Machining Center THINC-API Release 1.6.0.0	LHuynh
06/27/2008	S5015-003-07	Public release for Machining Center THINC-API Release 1.6.3.0	LHuynh
07/25/2008	S5015-003-08	Public release for Machining Center THINC-API Release 1.6.4.0	LHuynh
10/27/2008	S5015-003-09	Public release for Machining Center THINC-API Release 1.7.00	LHuynh
12/18/2008	S5015-003-010	Public release for Machining Center THINC-API Release 1.8.0.0	LHuynh
04/30/2009	S5015-003-011	Public release for Machining Center THINC-API Release 1.9.1.0	LHuynh
10/12/2009	S5015-003-012	Public release for Machining Center THINC-API Release 1.10.0.0	LHuynh
4/28/2010	S5015-003-013	Public release for Machining Center THINC-API Release 1.11.0.0	LHuynh
09/15/2010	S5015-003-014	Public release for Machining Center	LHuynh

		THINC-API Release 1.11.1.0	
12/15/2010	S5015-003-015	Public release for Machining Center THINC-API Release 1.12.0.0	LHuynh
02/02/2011	S5015-003-016	Public release for Machining Center THINC-API Release 1.12.1.0	LHuynh
11/16/2011	S5015-003-017	Beta release for Machining Center THINC-API Release 1.14.0.0	LHuynh
01/15/2012	S5015-003-018	Beta release for Machining Center THINC-API Release 1.14.1.0	LHuynh
09/21/2012	S5015-003-019	Beta release for Machining Center THINC- API Release 1.14.2.0	LHuynh
12/04//2012	S5015-003-020	Beta release for Machining Center THINC-API Release 1.15.0.0	LHuynh

Table of Contents

Release Notes For Machining Center

1. Introduction

1.1 Disclaimer of Warranty

Okuma America Corporation makes no representations or warranties, either expressed or implied, by or with respect to anything in this document, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose or for any indirect, special or consequential damages.

1.2 Purpose

The purpose of the *Release Notes* document is to communicate major new features and changes in this release of the THINC-API for Machining Center libraries. It also documents known problems and workarounds.

1.3 Scope

This document describes Release 1.15.0.0 of THINC-API.

1.4 Definitions, Acronyms, and Abbreviations

GAC – Global Assembly Cache Windows folder located in 'C:\WINDOWS\assembly'

1.5 References

None

2. About This Release

Public Release 1.15.0.0 of the THINC-API library for Machining Center supports the following:

From this release, THINC API libraries will check dependency libraries during installation and at run-time. THINC API will fail to install or load if version of dependency libraries cannot support current version of THINC API.

Version of Okuma.CMDATAPI.dll in Public Release 1.15.0.0 is 2.5.0.0

Version of Okuma.CMCMDAPI.dll in Public Release 1.15.0.0 is 2.3.0.0

Version of APINotifierService.exe in Public Release 1.15.0.0 is 1.15.0.0

Version of Okuma.Flexnet.dll in Public Release 1.15.0.0 is 1.3.0.0

Version of Okuma.Apilog.dll in Public Release 1.15.0.0 is 1.4.0.0

The following functions of Okuma.CMDATAPI.dll library will be only available based on the version of NC control software:

Class	Interfaces
CTools2	Int32 GetMaxTools()
CTools2	Int32 GetToolNo(Int32 intPotNo)
CTools2	Int32 GetMaxPots()
CTools2	String* GetGroupNo(Int32 intPotNo)
CTools2	Int32 GetSerialNo(Int32 intPotNo)
CTools2	String* GetToolName(Int32 intPotNo)
CTools2	String* GetToolKind(Int32 intPotNo)
CTools2	Boolean IsToolInUse(Int32 intPotNo) void SetToolInUse(Int32 intPotNo, Boolean bInValue)
CTools2	Boolean IsStandardTool(Int32 intPotNo) void SetStandardTool(Int32 intPotNo, Boolean bInValue)
CTools2	Boolean IsAdjustmentTool(Int32 intPotNo) void SetAdjustmentTool(Int32 intPotNo, Boolean bInValue)
CTools2	CarrierStatusEnum GetCarrierStatus(Int32 intPotNo) ;
CTools2	ToolLifeModeEnum GetMode(Int32 intPotNo) void SetMode(Int32 intPotNo, ToolLifeModeEnum enValue)
CTools2	ToolLifeStatusEnum GetStatus(Int32 intPotNo)
CTools2	Int32 GetToolLife(Int32 intPotNo) void SetToolLife(Int32 intPotNo, Int32 intValue)
CTools2	Int32 GetToolLifeRemaining(Int32 intPotNo) void SetToolLifeRemaining(Int32 intPotNo, Int32 intValue)
CTools2	Int32 GetToolLifeRemainingTimeSecond(Int32 intPotNo)
CTools2	Double GetToolLengthOffset1(Int32 intPotNo) ArrayList* GetToolLengthOffset1(Int32 intFromPotIndex, Int32 intToPotIndex) void SetToolLengthOffset1(Int32 intPotNo, Double dblValue) void AddToolLengthOffset1(Int32 intPotNo, Double dblValue) void CalToolLengthOffset1(Int32 intPotNo, Double dblValue)
CTools2	Double GetToolLengthOffset2(Int32 intPotNo) ArrayList* GetToolLengthOffset(Int32 intFromPotIndex, Int32 intToPotIndex) void SetToolLengthOffset2(Int32 intPotNo, Double dblValue) void AddToolLengthOffset2(Int32 intPotNo, Double dblValue) void CalToolLengthOffset2(Int32 intPotNo, Double dblValue)
CTools2	Double GetToolLengthOffset3(Int32 intPotNo) ArrayList* GetToolLengthOffset3(Int32 intFromPotIndex, Int32 intToPotIndex) void SetToolLengthOffset3(Int32 intPotNo, Double dblValue) void AddToolLengthOffset3(Int32 intPotNo, Double dblValue) void CalToolLengthOffset3(Int32 intPotNo, Double dblValue)

CTools2	Double GetCutterRCompOffset1(Int32 intPotNo) void SetCutterRCompOffset1(Int32 intPotNo, Double dblValue) void AddCutterRCompOffset1(Int32 intPotNo, Double dblValue) ArrayList* GetCutterRCompOffset1(Int32 intFromPotIndex, Int32 intToPotIndex)
CTools2	Double GetCutterRCompOffset2(Int32 intPotNo) void SetCutterRCompOffset2(Int32 intPotNo, Double dblValue) void AddCutterRCompOffset2(Int32 intPotNo, Double dblValue) ArrayList* GetCutterRCompOffset2(Int32 intFromPotIndex, Int32 intToPotIndex)
CTools2	Double GetCutterRCompOffset3(Int32 intPotNo) void SetCutterRCompOffset4(Int32 intPotNo, Double dblValue) void AddCutterRCompOffset3(Int32 intPotNo, Double dblValue) ArrayList* GetCutterRCompOffset3(Int32 intFromPotIndex, Int32 intToPotIndex)
CTools2	Double GetToolLengthWearOffset(Int32 intPotNo) void SetToolLengthWearOffset(Int32 intPotNo, Double dblValue) void AddToolLengthWearOffset(Int32 intPotNo, Double dblValue) ArrayList* GetToolLengthWearOffset(Int32 intFromPotIndex, Int32 intToPotIndex)
CTools2	Double GetCutterRCompWearOffset(Int32 intPotNo) void SetCutterRCompWearOffset(Int32 intPotNo, Double dblValue) void AddCutterRCompWearOffset(Int32 intPotNo, Double dblValue) ArrayList* GetCutterRCompWearOffset(Int32 intFromPotIndex, Int32 intToPotIndex)
CTools2	ToolTypeEnum GetToolType(Int32 intPotNo) void SetToolType(Int32 intPotNo, ToolTypeEnum enValue) ArrayList* GetToolType(Int32 intFromPotIndex, Int32 intToPotIndex)
CTools2	Double GetToolAngle(Int32 intPotNo) void SetToolAngle(Int32 intPotNo, Double dblValue) void AddToolAngle(Int32 intPotNo, Double dblValue) ArrayList* GetToolAngle(Int32 intFromPotIndex, Int32 intToPotIndex)
CTools2	Double GetToolDiameter(Int32 intPotNo) void SetToolDiameter(Int32 intPotNo, Double dblValue) void AddToolDiameter(Int32 intPotNo, Double dblValue) ArrayList* GetToolDiameter(Int32 intFromPotIndex, Int32 intToPotIndex)
CTools2	Double GetToolNoseDiameter(Int32 intPotNo) void SetToolNoseDiameter(Int32 intPotNo, Double dblValue) void AddToolNoseDiameter(Int32 intPotNo, Double dblValue) ArrayList* GetToolNoseDiameter(Int32 intFromPotIndex, Int32 intToPotIndex)

The following functions of Okuma.CMCMDAPI.dll library will be only available based on the version of NC control software:

Class	Interfaces
CATC2	Sub UnRegisterToolPot(ByVal intPotNo As Integer, Optional ByVal intToolNo As Integer = 0, Optional ByVal blnRandomATC As Boolean = False)
CATC2	RegisterToolPot(ByVal intToolNo As Integer, ByVal intPotNo As Integer, ByVal intGroupNo As Integer, ByVal intSerialNo As Integer, Optional ByVal blnDummyTool As Boolean = False, Optional ByVal strToolName As String = "")
CATC2	RegisterToolPot(ByVal intPotNo As Integer, ByVal intGroupNo As Integer, ByVal intSerialNo As Integer, Optional ByVal blnDummyTool As Boolean = False, Optional ByVal strToolName As String = "")
CATC2	SetToolCarrierStatus(ByVal intPotNo As Integer, ByVal enCarrierStatus As CarrierStatusEnum)
CATC2	SetToolKind(ByVal intPotNo As Integer, ByVal enToolKind As ToolKindEnum)
CATC2	SetToolLifeStatus(ByVal intPotNo As Integer, ByVal enToolLifeStatus As ToolLifeStatusEnum)

The following functions of Okuma.CMDATAPI.dll library will be only available in OSP-P200 control machines:

Classes	Interfaces
CIO	OnOffStateEnum GetUserTaskIOVariable(IOTypeEnum enIO, Int32 intIndex) ;
CIO	void SetUserTaskOutputVariable(Int32 intIndex, OnOffStateEnum enValue) ;
CIO	OnOffStateEnum GetProtectedUserTaskOutputVariable(Int32 intIndex) ;
CIO	void SetProtectedUserTaskOutputVariable(Int32 intIndex, OnOffStateEnum enValue) ;
CTools2	All functions in CTools2 class

The following functions of Okuma.CMCMDAPI.dll library will be only available in OSP-P200 control machines:

Classes	Interfaces
CATC2	All functions in CATC2 class
CMachine	SetUserAlarm(ByVal enAlarm As UserAlarmEnum, Optional ByVal strAlarmMessage As String = "")
CMachine	ClearUserAlarmD()

Note: User Alarm will require OKUMA.MC.UserAlarm license feature in order to function on OSP-P200 machine if machine can support.

For questions about these functions, please consult with your Okuma representatives.

3.   New Features

3.1.1   *Command API*

Classes	Interfaces
CMachine	SetUserAlarm(ByVal enAlarm As UserAlarmEnum, Optional ByVal strAlarmMessage As String = "")
CMachine	ClearUserAlarmD()

3.1.2   *Data API*

Classes	Interfaces
CMachine	GetNCStatus(NCStatusEnum enNCStatus)
CMachine	GetCurrentAlarmMessage()
CProgram	CycleComplete()
CProgram	ScheduleProgramCycleComplete()
CProgram	GetActiveScheduleProgramFileName()
CSpec	Boolean GetPLCSpecCode(Int32 intPLCSpecCodeIndex, Int32 intBit)
CSpec	Boolean GetPLCSpecCode2(Int32 intPLCSpecCodeIndex, Int32 intBit)

4.   Known Bugs and Limitations

This section identifies known and existing problems in Release 1.15.0.0 and describes any work-arounds.

4.1   Defect

4.2   General Defect

DATA-API library, Okuma.CMDATAPI.dll, cannot create directly under VS2003 ASP.NET web application.

Solution/Work-arounds: Create and initialize DATA-API in a separate thread. All function calls must be called from objects created inside separated thread mentioned above.

DATA-API library can only support applications designed with single-threaded apartment of COM threading model.

Solution/Work-arounds: None

4.2.1   *Data-API*

4.2.1.1 CAxis class

Function: AxisTypeEnum GetAxisType (AxisIndexEnum enValue)
    Symptom: Failed to access machine data for YS axis. YS axis is available in NC-HMI.

    Solution/Work around: None.

    All functions having YS axis as input parameter are affected by GetAxisType function and fail to get/set NC data.

Function: String* GetAxisName (AxisIndexEnum enValue)

Symptom: Failed to access machine data for YS axis. YS axis is available in NC-HMI.

Solution/Work around: None.

4.2.1.2 CMOPTool class

Function: GetToolDataNumber(ByVal *intToolNo* As Integer, ByVal *intClassNo* As Integer) As Integer
Symptom: This function cannot get tool data number given tool number greater than 300.
Solution/Work around: None.

4.2.1.3 CSpindle class

Function: Double SpindleStateEnum GetSpindleState()
Symptom: Function return incorrect state - When machine is in Manual mode, the spindle is set to ORIENTATION mode.
NC-HMI and THINC-API library returns the same state. Then, machine mode is changed to either AUTO or MDI, NC_HMI still report the same state as before, but THINC-API library reports spindle in stop mode.

Solution/Work around: None

4.3   Defects Fixed in this Release

4.3.1   CIO Class

4.3.1.1 Int32 GetWordIO(IOTypeEnum enIO, Int32 intAddressIndex)
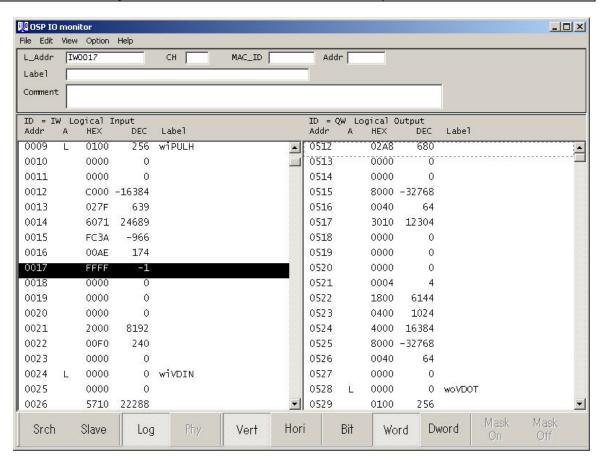
Function: Int32 GetWordIO(IOTypeEnum enIO, Int32 intAddressIndex)

Symptom: Function does not represent data correctly even the data has the same result from OSP IO Monitor application. The functions should return positive values only to correctly represent 2 to the power of N bits.
OSP IO Monitor application will return negative values when the actual data is greater than 32767.

For example: if bit 15 of word index 515 is ON then function will return 32768 meanwhile OSP IO Monitor displays -32768.
Or if all bits of word index 17 are ON then function will return 65535 meanwhile OSP IO Monitor displays -1.

4.3.2 CTool Class

Function: GetStatus()
Sympton: The function returns ToolLifeStatusEnum::NG2_WEAR instead of ToolLifeStatusEnum::NG1_WEAR when the condition is met.