Architectural Specification for Adaptive Panel Bending: Proprietary Instruction Sets, Real-Time Control Architectures, and HMI Visualization

1. Executive Summary

This comprehensive research report outlines the software architecture required to transition from legacy, deterministic CNC control paradigms to a stochastic, adaptive control model suitable for Industry 4.0 Panel Bending machinery. The target system utilizes a Beckhoff TwinCAT 3 environment to drive a high-performance Panel Bender (comparable to Salvagnini P4/P2 classes) capable of Automatic Blankholder Adjustment (ABA) and real-time Material Attitude Correction (MAC 2.0).

The central thesis of this architecture is that standard ISO 6983 (G-code) is fundamentally insufficient for modern sheet metal forming due to its inability to encapsulate material rheology, dynamic tooling configurations, and non-linear adaptive logic. To address this, we define a proprietary **Machine Instruction Set** utilizing a dual-schema approach: an XML-based **Job Definition Object (JDO)** for robust PLC ingestion, and a JSON-based **Visualization Data Model (VDM)** for high-performance web-based HMI rendering.

The report details the integration of these schemas with the TwinCAT Real-Time Kernel, specifically leveraging the TwinCAT ADS (Automation Device Specification) protocol for variable injection. This allows the control system to modify motion trajectories within the millisecond-range processing loop, compensating for material springback and thickness variability dynamically.[1] Furthermore, the report specifies the data structures required to drive a "Digital Twin" HMI, utilizing vertex shaders and scene graphs to visualize physical deformation in real-time, moving beyond static animation to true process simulation.[4]

---

2. The Obsolescence of ISO 6983 (G-Code) in Adaptive Panel Bending

The persistence of G-code in the manufacturing sector is largely due to inertia rather than suitability for modern applications. While adequate for chip-cutting processes (milling, turning) where the workpiece is treated as a rigid solid and tool paths are absolute, G-code fails catastrophically when applied to the non-linear, elastic-plastic deformation processes

inherent in Panel Bending.[6]

2.1 The Deterministic vs. Stochastic Conflict

G-code is an imperative language; it dictates explicit axis coordinates (e.g., G01 X100 Y50 Z-2.0). This assumes that reaching the coordinate Z-2.0 guarantees the desired geometric outcome. In sheet metal bending, this assumption is false. The final angle of a bend is a function of the tool position, the material thickness, the tensile strength, the rolling direction (grain), and the resulting springback.[8]

- **Legacy Approach:** A programmer calculates a "Springback Allowance" in the CAM software and hard-codes the target to Z-2.5. If the material batch changes (e.g., higher tensile strength), the part is under-bent, and the G-code program must be rewritten or a global offset applied manually by an operator.[7]
- **Adaptive Requirement:** The control system requires a *Target Angle* (90.0°), not a target axis position. The machine must determine the axis position dynamically during the cycle by measuring force and deflection.[1] G-code has no syntax for "Bend to 90°, adjusting for live force feedback." It separates the "What" (geometry) from the "How" (process), but panel bending requires the "How" to be derived in real-time.

2.2 Complexity of Masked-Time Tooling (ABA)

Modern panel benders feature Automatic Blankholder Adjustment (ABA), where the upper tool is not a single solid block but a composite tool assembled from varying segment widths (e.g., 5mm, 10mm, 25mm, 100mm).[12] This setup changes dynamically between bends to accommodate different flange lengths and clearance requirements.

- **G-Code Deficiency:** G-code uses a T word (e.g., T01) to select a pre-defined tool. It lacks the data structure to describe a "Tool Composition" where Tool_Current =. Attempting to map every possible permutation of tool segments to a discrete T-code would result in thousands of definitions, making the controller logic unmanageable.[12]
- **Set-Theoretic Logic:** ABA requires a set-theoretic description of the tool station: "Select segments A, B, and C; gap them by 50mm." This requires array handling and logical conditional processing (e.g., "If Segment A is stuck, try Segment B"), which is extremely cumbersome in the linear execution flow of G-code macros.[15]

2.3 Hierarchical State Management vs. Linear Streams

A Panel Bending Job is inherently hierarchical:

1. **Job Level:** Material properties (K-Factor, Thickness), Global Safety Zones.
2. **Part Level:** Geometric definitions, Cycle time targets.
3. **Sequence Level:** Rotations, Flips, Handling operations.
4. **Bend Level:** The specific forming operation, containing sub-states (Approach, Clamp, Form, Decompress, Return).

5. **Motion Level:** The interpolation of axes (X, Y, Z, C, Blades).

G-code flattens this rich hierarchy into a unidimensional stream of characters.[6] This loss of structure means the PLC cannot easily query "What is the target angle of Bend 4?" without parsing the entire file from the start. In an adaptive system, the controller needs random access to any bend's parameters to apply corrections or skip steps based on sensor data.[17] An XML or JSON object model preserves this hierarchy, enabling the controller to treat the process as a structured database of operations rather than a tape to be played.[18]

2.4 Lack of Material Rheology Metadata

Adaptive technologies like MAC 2.0 rely on comparing *measured* behavior against *theoretical* models.[2] To do this, the controller needs to know the material's theoretical properties: Young's Modulus, Poisson's Ratio, and Nominal Thickness. G-code has no standard standardized mechanism to convey this metadata. While comments or custom variables (#500=2.0) could be used, they are non-standard, fragile, and lack type safety.[10] A proprietary schema allows these physical constants to be strongly typed attributes of the Job object, ensuring the physics engine in the PLC has the necessary boundary conditions to compute corrections.[20]

---

3. Proprietary Command Language Architecture: The "PB-XML" Schema

To overcome the limitations of G-code, we define a proprietary **Machine Instruction Set** encoded in XML. XML is selected for the PLC ingestion layer due to its robust validation capabilities (XSD) and native support within the Beckhoff TwinCAT environment via the Tc2_XmlDataSrv library.[22]

The schema, designated **PB-XML (Panel Bend Extensible Markup Language)**, serves as the authoritative definition of the manufacturing process. It is generated by the Post-Processor from the BendSequence object and consumed by the TwinCAT PLC.

3.1 Global Job Context and Header

The root of the schema defines the administrative and physical context of the job. Unlike G-code headers which are typically just comments, these fields are active variables loaded into the PLC's process image.

```
XML
```

```
<PanelBendJob SchemaVersion="3.0" xmlns="http://schemas.proprietary.com/pb/2026">
  <Header>
    <JobID>JOB-2026-X99-ALPHA</JobID>
    <CreationDate>2026-02-04T14:30:00Z</CreationDate>
    <Author>PostProcessor_v4.2</Author>
    <ProductionTarget>
      <Quantity>500</Quantity>
      <CycleTimeLimit Units="s">45.0</CycleTimeLimit>
    </ProductionTarget>
    <MachineCompatibility>
      <TargetModel>P4-2516</TargetModel>
      <MinFirmwareVersion>4.1.2.90</MinFirmwareVersion>
    </MachineCompatibility>
  </Header>
```

Architectural Insight: The MachineCompatibility block allows the PLC to reject jobs created for incompatible hardware configurations before attempting execution, preventing physical crashes—a safety feature absent in standard G-code.[24]

3.2 Material Physics Definition (The "Physics Engine" Input)

This section is critical for the MAC 2.0 adaptive control. It defines the "Digital Twin" of the raw material.

XML

```
  <MaterialDefinition>
    <MaterialID>AL-5052-H32</MaterialID>
    <Physics>
      <NominalThickness Units="mm">2.00</NominalThickness>
      <ThicknessTolerance Units="mm">0.05</ThicknessTolerance>
      <ElasticModulus Units="GPa">70.0</ElasticModulus>
      <YieldStrength Units="MPa">215</YieldStrength>
      <PoissonRatio>0.33</PoissonRatio>
    </Physics>
    <ProcessParameters>
      <DesignKFactor>0.42</DesignKFactor>
      <GrainOrientation>Longitudinal</GrainOrientation>
      <SurfaceFrictionCoeff>0.15</SurfaceFrictionCoeff> </ProcessParameters>
  </MaterialDefinition>
```

Architectural Insight: The ElasticModulus and YieldStrength are not used for motion planning directly but are inputs for the **Force-Deflection Observer** in the PLC. When the blade contacts the sheet, the PLC compares the real-time torque rise against a theoretical curve derived from these values. A deviation indicates a material inconsistency, triggering the MAC 2.0 compensation logic.[1]

3.3 Dynamic Tooling Configuration (The ABA Model)

This schema block replaces the "Tool Change" command. It describes the *configuration* of the modular tool.

XML

```
<ToolingSetup>
  <BlankHolder>
    <ConfigStrategy>Automatic</ConfigStrategy>
    <TotalLength Units="mm">1200</TotalLength>
    <SegmentStack>
      <Segment ID="Fixed_Left" Width="100" />
      <Segment ID="Mobile_A" Width="50" Count="4" />
      <Segment ID="Mobile_B" Width="25" Count="2" />
      <Segment ID="Gap_Spacer" Width="50" IsActive="false" /> <Segment ID="Fixed_Right" Width="100" />
    </SegmentStack>
    <SetupTimeEstimate Units="s">12.5</SetupTimeEstimate>
  </BlankHolder>
  <AuxiliaryTools>
    <Blade ID="Neg_Blade_Offset" Type="Negative" Offset="3.0" />
  </AuxiliaryTools>
</ToolingSetup>
```

Architectural Insight: The SegmentStack is parsed by the PLC into an array of axis target positions. In the P4 architecture, electric actuators slide these segments. The PLC iterates through this list, calculates the cumulative position of each segment carrier, and issues synchronized motion commands to the ABA axes.[1][2]

3.4 The Bend Sequence (Event-Driven Process Definition)

This is the core execution logic. It uses an object-oriented approach where each Step is a

self-contained unit with properties, error handlers, and specific motion profiles.

XML

```
<ProcessSequence>
  <Step ID="1" Type="Reference">
    <Parameters>
      <Side>X_Negative</Side>
      <PushForce Units="N">50</PushForce> </Parameters>
  </Step>

  <Step ID="2" Type="Bend" BendID="B1">
    <Geometry>
      <TargetAngle>90.0</TargetAngle>
      <FlangeLength>25.4</FlangeLength>
      <InternalRadius>2.0</InternalRadius>
    </Geometry>

    <MotionProfile>
      <ApproachSpeed>100</ApproachSpeed> <BendingSpeed>10</BendingSpeed>
<ReturnSpeed>100</ReturnSpeed>
      <DwellTime Units="ms">150</DwellTime>
    </MotionProfile>

    <AdaptiveControl>
      <Mode>MAC2_0_Full</Mode> <CorrectionStrategy>
        <ThicknessComp>true</ThicknessComp>
        <SpringbackComp>true</SpringbackComp>
      </CorrectionStrategy>
      <SafetyLimits>
        <MaxOverbend>3.0</MaxOverbend> </SafetyLimits>
    </AdaptiveControl>
  </Step>

  <Step ID="3" Type="Manipulate">
    <Action>Rotate</Action>
    <Angle>90.0</Angle>
    <Interpolation> Blended </Interpolation>
  </Step>
</ProcessSequence>
```

```
</PanelBendJob>
```

Architectural Insight: The AdaptiveControl block allows per-bend granularity. Critical bends can have MAC2_0_Full enabled (slower, high precision), while non-critical bends (e.g., hems) can use MonitorOnly for speed. This flexibility is impossible in global G-code modal switches.[1]

---

4. Real-Time Adaptive Control Architecture (MAC 2.0 Integration)

The XML defines the "Intent." The "Execution" relies on the **Variable Injection** mechanism. This architecture enables the controller to modify the trajectory of the machine *while the tool is in motion*, based on high-frequency sensor data.

4.1 Physics of Material Attitude Correction (MAC 2.0)

The MAC 2.0 system functions as a real-time feedback loop. It does not simply measure the final angle; it measures the material's resistance to deformation.[1]

1. **Phase 1: Touch & Detect:** The bending blade approaches the sheet. The servo drive torque monitoring detects the spike in current corresponding to physical contact. The encoder position at this moment establishes the precise material thickness, overwriting the NominalThickness from the XML.[2]
2. **Phase 2: Elastic Sampling:** As the blade pushes the material through the elastic region (before permanent deformation), the system records the Force vs. Deflection curve.
3. **Phase 3: Stiffness Calculation:** The slope of this curve (dF/dx) represents the material's stiffness. The PLC compares this to the ElasticModulus defined in the Job File.
4. **Phase 4: Springback Prediction:** Using a proprietary algorithm (Neural Network or Lookup Table stored in TwinCAT), the system predicts the springback angle ($\Delta\theta$) for this specific bend.[20]
5. **Phase 5: Trajectory Injection:** The calculated $\Delta\theta$ is added to the target position, and the motion profile is updated on the fly.

4.2 PLC Data Structures & Memory Layout

To facilitate this high-speed interaction between the high-level logic (C#/.NET) and the low-level control (PLC/NC), we define specific IEC 61131-3 Data Structures.

We utilize the {attribute 'pack_mode' := '1'} pragma to ensure tight byte alignment. This is critical for **TwinCAT ADS** communication, as it ensures that the memory layout in the C# application matches the PLC memory bit-for-bit, preventing marshalling errors.[25]

4.2.1 The Injection Structure (PLC)

Delphi

```
{attribute 'pack_mode' := '1'}
TYPE ST_AdaptiveBendContext :
STRUCT
    (* --- Inputs from Job File (XML) --- *)
    fNominalAngle      : LREAL; (* Target Angle in Degrees *)
    fDesignKFactor     : LREAL;
    fSheetThickness    : LREAL;
    fMaterialModulus   : LREAL;

    (* --- Real-Time Sensor Feedback --- *)
    fMeasuredTorque    : LREAL; (* Nm, from Drive *)
    fMeasuredContactPos : LREAL; (* mm, Encoder position at contact *)

    (* --- Calculated Adaptive Variables --- *)
    fComputedThickness  : LREAL; (* Derived from ContactPos *)
    fStiffnessDeviation : LREAL; (* % diff from nominal *)
    fSpringbackCorrection: LREAL; (* Degrees *)
    fCrowningOffset    : LREAL; (* mm, center deflection comp *)

    (* --- Control Flags --- *)
    bInjectionActive   : BOOL;  (* True when ext. logic is overriding *)
    eMAC_State         : E_MAC_State; (* IDLE, SAMPLING, CALCULATING, CORRECTING *)
END_STRUCT
END_TYPE
```

4.3 The Injection Pipeline (TwinCAT ADS)

The injection pipeline bridges the non-real-time world (HMI, Database, Cloud Analytics) and the real-time world (PLC, NC).

4.3.1 ADS Sum Commands for Latency Reduction

In standard ADS communication, reading/writing variables sequentially introduces round-trip latency (Network Overhead). For MAC 2.0, we need to read sensor data and write correction factors within a single PLC cycle (< 1ms). We employ **ADS Sum Commands**.[28]

- **Mechanism:** The HMI/Controller bundles multiple Read/Write requests into a single TCP/IP packet.

- **Application:**
  1. **Read Sum:** Get MeasuredTorque, Position, Status (1 call).
  2. **Compute:** C# or C++ Algorithm calculates the SpringbackCorrection.
  3. **Write Sum:** Inject SpringbackCorrection, CrowningOffset (1 call).
- **Performance:** This reduces communication time from ~10ms (multiple calls) to ~0.5ms (single sum call), enabling 1kHz control loops even from a Windows-level application.[31]

4.3.2 PLC Real-Time Injection Logic (Structured Text)

The PLC code running in the MotionTask (High Priority) handles the injection.

Delphi

```
// Running in cyclic Motion Task (e.g., 250us cycle)
IF GVL.CurrentBend.bAdaptiveActive THEN

    CASE GVL.CurrentBend.eMAC_State OF

        E_MAC_State.SAMPLING:
            // Capture torque data during elastic deformation
            fbDataLog(Input := GVL_IO.Axis_Blade.fTorque);
            IF GVL_IO.Axis_Blade.fPosition > fYieldPointThreshold THEN
                GVL.CurrentBend.eMAC_State := E_MAC_State.CALCULATING;
            END_IF

        E_MAC_State.CALCULATING:
            // Calculate correction (or wait for C++ module to do it)
            // Simplified Formula:
            fDelta := (fbDataLog.AvgTorque - fNominalTorque) * fMaterialFactor;
            GVL.CurrentBend.fSpringbackCorrection := fDelta;

            // INJECT: Update the NC Axis Target Position immediately
            // This overrides the original XML target
            fbMoveAbs.Position := GVL.CurrentBend.fNominalAngle + fDelta;
            fbMoveAbs.Execute := TRUE; // Retrigger motion

            GVL.CurrentBend.eMAC_State := E_MAC_State.CORRECTING;

    END_CASE
END_IF
```

Architectural Insight: This logic demonstrates the system's ability to "change its mind" mid-motion. The fbMoveAbs is re-triggered with a new target position derived from the physics calculation, seamless to the observer but critical for accuracy.[32]

---

5. HMI Visualization Data Structure: The Digital Twin

The HMI for an adaptive machine serves a different purpose than a standard CNC HMI. It must visualize **Process** (Deformation), not just **Position** (Machine coordinates). We utilize a **WebGL-based** visualization (integrated into TwinCAT HMI via HTML5/JavaScript) driven by a JSON data stream.

5.1 Visualization Architecture Overview

- **Renderer:** Three.js or Babylon.js, hosted in a TwinCAT HMI TcHmiHtmlHost control.[34]
- **Data Source:** TwinCAT ADS Web Socket (Secure ADS).
- **Update Rate:** 30-60Hz.

5.2 Scene Graph & Data Binding

The 3D environment is structured as a Scene Graph, where nodes represent machine components. A JSON configuration file maps these nodes to PLC variables.

JSON Schema for Visualization (VDM):

JSON

```
{
  "SceneConfiguration": {
    "Camera": { "FollowTarget": "Blade_Center", "Distance": 1500 },
    "Lighting": { "Type": "Industrial", "Intensity": 1.2 }
  },
  "ComponentMap":
}
```

5.3 Real-Time Deformation Rendering (Vertex Shaders)

The most challenging requirement is visualizing the sheet metal bending. A simplistic approach of loading a new STL for every 1-degree increment is computationally prohibitive

and causes visual stuttering. We implement a **Parametric Vertex Shader** solution.[5]

Mechanism:

1. **Static Mesh:** The HMI loads the *flat* sheet metal mesh once at the start of the job.
2. **Uniforms:** The HMI passes dynamic variables (Uniforms) to the GPU shader:
   - u_BendLineOrigin (Vector3): The geometric center of the current bend.
   - u_BendLineNormal (Vector3): The axis of rotation.
   - u_CurrentAngle (Float): Linked directly to the PLC axis position (fbBendUnit.fActPos).
3. **Vertex Logic (GLSL):** The GPU calculates the rotation of every vertex *past* the bend line in real-time.

Conceptual GLSL Shader Code:

OpenGL Shading Language

```
attribute vec3 position;
uniform float u_CurrentAngle; // From PLC
uniform vec3 u_BendOrigin;
uniform vec3 u_BendAxis;

void main() {
  vec3 vPos = position;

  // Check if vertex is in the "Active Flange" zone
  // This is a simplified planar check
  float distanceToLine = dot(vPos - u_BendOrigin, cross(u_BendAxis, vec3(0,1,0)));

  if (distanceToLine > 0.0) {
    // Apply Rotation Matrix
    // The vertex "bends" purely on the GPU, zero CPU cost
    vPos = rotate(vPos, u_BendOrigin, u_BendAxis, u_CurrentAngle);
  }

  gl_Position = projectionMatrix * modelViewMatrix * vec4(vPos, 1.0);
}
```

Architectural Insight: This allows the HMI to visualize the MAC 2.0 corrections visually. If the PLC injects a +2.5° springback correction, the 3D model on screen immediately "overbends" by 2.5° because u_CurrentAngle reflects the physical axis position. This provides immediate,

intuitive feedback to the operator.[37]

5.4 Handling Large Array Data (ABA Segments)

Visualizing the ABA tool requires updating the positions of potentially 30-50 individual tool segments. Sending 50 separate ADS requests is inefficient.

Optimization Strategy:

1. **PLC:** Define a single large array: aToolPos : ARRAY [1..50] OF REAL.
2. **ADS:** Use AdsStream to read the entire memory block (200 bytes) in one call.[39]
3. **JS/WebGL:** Parse the Float32Array and update the position.x of the 50 segment meshes in the 3D scene loop. This ensures the tool setup animation is fluid and synchronous with the real machine setup time.[41]

---

6. Implementation Strategy & Systems Integration

6.1 System Architecture Diagram

The architecture is defined by three distinct layers:

Layer	Technology	Function	Cycle Time
HMI / Client	HTML5 / WebGL / JSON	3D Visualization, User Input, Job Loading	30-50 ms (30Hz)
Middleware	C#.NET / ADS Sum Commands	XML Parsing, Data Marshalling, DB Logging	1-10 ms
Control Kernel	TwinCAT 3 PLC / NC (C++)	Motion Planning, MAC 2.0 Injection, Safety	0.25 - 1 ms

6.2 Beckhoff Library Dependencies

Successful implementation relies on specific Beckhoff libraries identified in the research:

- **Tc2_XmlDataSrv (TF6421):** For parsing the PB-XML Job files directly in the PLC runtime.[22]
- **Tc2_MC2 (TF5000):** For basic PTP motion control of the manipulator and blades.[33]

- **Tc3_Math:** For statistical functions (Regression) used in the MAC 2.0 curve fitting.
- **Tc3_HMI (TE2000):** For hosting the WebGL visualization framework.[41]
- **TwinCAT ADS.NET:** For the high-performance injection pipeline.[27]

---

7. Conclusion

This report defines a comprehensive software architecture for a next-generation Panel Bender. By rejecting the legacy constraints of G-code and adopting a proprietary **XML/JSON Instruction Set**, the system gains the ability to describe the complex, non-linear realities of sheet metal forming.

The integration of **Variable Injection** via TwinCAT ADS enables the **MAC 2.0** adaptive control loop, transforming the machine from a passive executor of coordinates to an active participant in process quality. The material is measured, modeled, and compensated for in real-time.

Finally, the **Parametric 3D HMI** architecture ensures that the operator's view of the process is not a static playback but a live, physics-driven simulation of the machine's actual behavior. This "Digital Twin" approach reduces setup time, enhances safety, and provides the transparency required for autonomous manufacturing cells.

Table 1: Comparative Analysis of Control Architectures

Feature	Legacy G-Code (ISO 6983)	Proposed Proprietary Architecture (PB-XML)
Structure	Linear, unstructured text stream	Hierarchical, Object-Oriented (Job -> Step -> Op)
Material Physics	Not supported (comments only)	Strong-typed attributes (Modulus, Yield, K-Factor)
Tooling	Fixed Index (T01)	Dynamic Composition (Set logic for ABA segments)
Adaptive Control	Impossible (Requires re-post)	Real-time Injection (MAC 2.0) via ADS
Visualization	Static Backplot of lines	Real-time Vertex

		Deformation (Digital Twin)
Variable Resolution	Compile-time (Post-Processor)	Run-time (PLC/Sensor Fusion)

This architecture satisfies all requirements for a high-end, adaptive Panel Bending solution suitable for Industry 4.0 integration.

Works cited

1. The productive, flexible bending solution., accessed February 4, 2026, https://www.italianmachines.eu/userfiles/catalog/pdf/7cd4f0c7f470d00702d4d440b82a248f.pdf
2. The flexible bending solution. - Sp-Tech, s.r.o., accessed February 4, 2026, https://www.sp-tech.cz/static/soubory/stranka-71/prospekt-salvagnini-p2-152.pdf
3. Panel Bending 4.0 | amplify | The Pepperl+Fuchs Magazine, accessed February 4, 2026, https://amplify.pepperl-fuchs.com/2018/panel-bending-4-0/
4. Bending Simulation Framework for Rapid Feasibility Checks of Sheet Metal Parts - CAD Journal, accessed February 4, 2026, https://www.cad-journal.net/files/vol_23/CAD_23(1)_2026_85-100.pdf
5. New 3D framework revolutionizes data visualization & simulation - M2, accessed February 4, 2026, https://m2dot.com/en/data-news/data-blog/article/new-3d-framework-revolutionizes-data-visualization-simulation
6. What is the "Detailed format classification" of the ISO 6983 (G-Code spec), accessed February 4, 2026, https://3dprinting.stackexchange.com/questions/6629/what-is-the-detailed-format-classification-of-the-iso-6983-g-code-spec
7. CNC Machining's Best Kept Secret: Parametric Programming - MultiCam Inc., accessed February 4, 2026, https://www.multicam.com/en/resources/blog/2017/cnc-machinings-best-kept-secret-parametric-programming
8. Sheet Metal Bending Process Card Standard: The Ultimate Guide - ADH Press Brake, accessed February 4, 2026, https://pressbrake.adhmt.com/sheet-metal-bending-process-card-standard/
9. What is Sheet Metal Bending - A comprehensive guide - Jiga, accessed February 4, 2026, https://jiga.io/articles/sheet-metal-bending/
10. Subprograms, Macros and Parametric Programming for CNC Machining - Thomasnet, accessed February 4, 2026, https://www.thomasnet.com/articles/custom-manufacturing-fabricating/subprograms-macros-and-parametric-programming-for-cnc-machining/
11. The next best compact panel bender, accessed February 4, 2026, https://www.simted.com/docs/PX/PX-BukumMerkezi.pdf

12. p4-benders.pdf - Empire Machinery, accessed February 4, 2026, https://www.empire-machinery.com/wp-content/uploads/2021/06/p4-benders.pdf
13. Salvagnini P2 User Manual | PDF - Scribd, accessed February 4, 2026, https://www.scribd.com/document/761106673/Salvagnini-P2-User-Manual
14. The automatic hybrid adaptive Panel Bender. - Sanson Machinery Group, accessed February 4, 2026, http://www.sansonmachinery.com/wp-content/uploads/2013/12/P4Xe-UK-3962020312.pdf
15. G code vs other programming language : r/Machinists - Reddit, accessed February 4, 2026, https://www.reddit.com/r/Machinists/comments/ff27cv/g_code_vs_other_programming_language/
16. CNC Machining: Macros, Subprograms, and Parametric Programming - Fictiv, accessed February 4, 2026, https://www.fictiv.com/articles/cnc-machining-macros-subprograms-and-parametric-programming
17. TF5xxx | TwinCAT 3 Motion | Beckhoff USA, accessed February 4, 2026, https://www.beckhoff.com/en-us/products/automation/twincat/tfxxxx-twincat-3-functions/tf5xxx-motion/
18. PLC Basic 1:TWINCAT 3 Beckhoff parse XML creating XML object - YouTube, accessed February 4, 2026, https://www.youtube.com/watch?v=Jh2ur_wcOeQ
19. The structure of JSON training data configuration files - Amazon Neptune, accessed February 4, 2026, https://docs.aws.amazon.com/neptune/latest/userguide/machine-learning-processing-training-config-file-structure.html
20. The Real-Time Prediction of Cracks and Wrinkles in Sheet Metal Forming According to Changes in Shape and Position of Drawbeads Based on a Digital Twin - MDPI, accessed February 4, 2026, https://www.mdpi.com/2076-3417/15/2/700
21. Computer simulation and experimental investigation of sheet metal bending using laser beam scanning - SMU, accessed February 4, 2026, https://www.smu.edu/-/media/site/lyle/rcam/publications/3-computer-simulation-and-experimental-investigation-of-she.pdf
22. FB_XmlSrvWrite - Beckhoff Information System - English, accessed February 4, 2026, https://infosys.beckhoff.com/content/1033/tcxmldatasrv/11416793355.html
23. TF6421 | TwinCAT 3 XML Server | Beckhoff USA, accessed February 4, 2026, https://www.beckhoff.com/en-us/products/automation/twincat/tfxxxx-twincat-3-functions/tf6xxx-connectivity/tf6421.html
24. TF6421 - TwinCAT 3 | XML Server - download - Beckhoff, accessed February 4, 2026, https://download.beckhoff.com/download/document/automation/twincat3/TF6421_TC3_XML_Server_EN.pdf
25. Value marshalling with ANYTYPE concept - Beckhoff Information System, accessed February 4, 2026,

https://infosys.beckhoff.com/content/1031/tcadsnetref/7312581771.html
26. Marshalling structure with array of structure in C# .NET 7.0 - Stack Overflow, accessed February 4, 2026, https://stackoverflow.com/questions/78007487/marshalling-structure-with-array-of-structure-in-c-sharp-net-7-0
27. ADS-DLL .NET Samples - TwinCAT 3 - download, accessed February 4, 2026, https://download.beckhoff.com/download/document/automation/twincat3/TwinCAT_ADS_NET_Samples_EN.pdf
28. ADS-Sum Command: Reading or writing several variables - Beckhoff Information System, accessed February 4, 2026, https://infosys.beckhoff.com/content/1033/tc3_adssamples_net/185258507.html
29. SumHandleWrite Class - Beckhoff Information System - English, accessed February 4, 2026, https://infosys.beckhoff.com/content/1033/tcadsnetref/7313704075.html
30. Read variables in Twincat 3 continously with the C# API - Stack Overflow, accessed February 4, 2026, https://stackoverflow.com/questions/62004586/read-variables-in-twincat-3-continously-with-the-c-sharp-api
31. PLC programming using TwinCAT 3 - ADS (Part 15/18) - YouTube, accessed February 4, 2026, https://www.youtube.com/watch?v=JZChSdU2LMc
32. PC-based Motion control: First steps with TwinCAT 3 Motion Control. | Beckhoff USA, accessed February 4, 2026, https://www.beckhoff.com/en-us/support/webinars/pc-based-motion-control-first-steps-with-twincat-3-motion-control.html
33. TwinCAT 3 Motion Control Tutorial (Part 1): Axis Setup & Basic Configuration - YouTube, accessed February 4, 2026, https://www.youtube.com/watch?v=NcdApTo2TEA
34. TwinCAT HMI 3D Animations - Control Works, accessed February 4, 2026, https://control-works.com/twincat-hmi-3d-animations/
35. TWİNCAT 3 HMI 3D - Reddit, accessed February 4, 2026, https://www.reddit.com/r/TwinCat/comments/1kb9fr4/twincat_3_hmi_3d/
36. A computational tool for the analysis of 3D bending-active structures based on the dynamic relaxation method - Gianmarco Cherchi, accessed February 4, 2026, https://www.gianmarcocherchi.com/pdf/comp_tool_3D_bending-active-structures.pdf
37. 3D Graphics Overview - WPF - Microsoft Learn, accessed February 4, 2026, https://learn.microsoft.com/en-us/dotnet/desktop/wpf/graphics-multimedia/3-d-graphics-overview
38. Using ViewPort 3D in WPF | (3D Graphics, the easier way) - YouTube, accessed February 4, 2026, https://www.youtube.com/watch?v=30c8r-t5ulo
39. Write array of struct using TwinCAT.Ads through C# application - Stack Overflow, accessed February 4, 2026, https://stackoverflow.com/questions/39086088/write-array-of-struct-using-twincat-ads-through-c-sharp-application
40. [Communication Drivers] Beckhoff Automation GmbH TwinCAT ADS/AMS Driver |

Pro-face by Schneider Electric, accessed February 4, 2026, https://www.proface.com/en/node/45291
41. TwinCAT HMI | Beckhoff USA, accessed February 4, 2026, https://www.beckhoff.com/en-us/products/automation/twincat-3-hmi/
42. Cookbook "How to implement an ADS client" - Beckhoff Information System - English, accessed February 4, 2026, https://infosys.beckhoff.com/content/1033/tcadscommon/12439477899.html