Differences MPE to MCE

The rational for the new MCE (Master Control Electronics):

- cost
- components obsolete
- core shared firmware is much larger

Firmware Changes

The firmware on CTP devices controls all aspects of the device operation.: media handling, plotting and all of the hardware connected to the device (debris removal, for example).

Architecture

Multiple parts to the firmware (Main code, Xilinx code, Boot code, Satellite main and boot code, POST (Power On Self Test) code. Each part has to be downloaded separately, a process made simple with Service Shell (**see Appendix A**).

Rationale:

- Xilinx chip can be changed without having to download new main firmware.
- Device architecture now has satellite boards (GENINE and PDB boards) that run their own main and boot code.
- Some problems on the MPE-based devices were caused by old boot code. With the new architecture, the boot code can be maintained so everything stays up to date.
- POST (Power On Self Test) code tests the board's hardware before starting the Main code

Standardization of Units of Measure

The CAU (Creo Arbitrary Units) are gone: Inches and mils are gone and drum positions in ticks are gone. MPE firmware uses imperial measurements (inches, mils) for most things. MCE firmware is metric across the board and uses real units. Carriage move commands, for example, are in mm (position) and mm/s (velocity) and drum move commands are in degrees.

Rationale:

- Metric provides a standard.
- The use of degrees for the drum means that any parameters using degrees will remain unchanged even if the encoder resolution is changed.

```
>>Carriage moveto <position in mm>

>>Drum move <position in degrees>
```

Cross reference table?

Commands and Parameters

For more details, see **Appendix B**.

1. **As soon as a parameter is changed, it is saved**.

Rationale:
- When a service engineer leaves site, the machine should be running with saved parameters – won't change functionality when device is next reset.
- Commands should exist for changing/calibrating most if not all parameters.

2. **PRM (for parameter), has replaced NVS in commands**, e.g., <<prm all>> instead of <<nvs dump>>

   Rationale:
   - During development, the NVS name was used for something else, so a new name was needed for the parameter subsystem.

3. **No <<nvs upgrade>> command** required during firmware upgrades

   Rationale:
   - New parameters automatically take their default values, so they don't need to be set manually in an upgrade command.
   - Any parameters that do need to be changed will be handled in an upgrade script instead of a firmware command. The script will be less error-prone than the firmware has been in the past.

4. **Full Load And Unload Sequences** – only need load and unload commands instead of the smaller pick, load, unload, process commands. **See Appendix C for more information**.

   Rationale:
   - Simpler command to get a plate onto the drum
   - Having the full load and unload sequences allows easier scheduling of media for full throughput
   - Load and unload is consistent across all products

5. **Disabling sensors is done using sensor name**, not <<set al>>. **See Appendix E** for list of sensors and more examples.

   For example to disable the table down sensor:
   MPE: set al uts 0

   MCE: scan al_tabledown bon

   Rationale:
   - Names are standardized, so there is more consistency across products
   - No need to have the long list of "set al" parameters
   - All sensors can be bypassed

6. Help <<command>> lists all combinations. **See Appendix G for examples.**
7. Verbose settings - **see Appendix H**
   For MPE, the command <<set verbose>>is used to control the display of information about a task (such as the picker or engine task). Level 0 corresponds to basic information; while 3 provides troubleshooting/debug information.

   For MCE firmware, the verbosity level is one of the following C, S, E, or D, which corresponds to Customer, Service, Engineer and Developer. The command, <<vbs user customer>>, gives the least amount of information The task name is a three-letter acronym.

Functionality

The MCE uses code from the new FW1 (Firmware 1) environment.

MCE firmware is more consistent across all devices.

8. **Concept of Release in response to a Suspend** (and Clear taken away)
   See Appendix D for more information.

   Rationale:
   - **Clear** was not well understood and didn't really help the operator.
   - Release tells the firmware to move actuators, etc. into a state where the media can be removed manually from the machine. The operator can then get the plate out of the system without damaging anything.

9. When the device is suspended, cannot move actuators.

   Rationale:
   - If actuators can be moved, the sequences get much more complicated because the hardware is in adifferent state than expected.
   - The release sequence takes care of freeing the plate

Appendix A – Service Shell

Will become the standard. Currently, Service Shell is not much more than a terminal program. It does have a few nice features:

- easy downloads using the “download” command
- automatic logging to log files (with time-stamping for each line)
- supports multiple connections to the same device
- includes splot for scripts

The future plans include the creation of “applets” – applications built into Service Shell that can do specific tasks. Examples of these include:

- automatic firmware upgrades/downgrades (including NVS upgrades)
- version checking to make sure all parts are compatible
- automatic system pressure logging

Appendix B - NVS parameters

There are many less parameters in MCE than in MPE. The reasons for this are two-fold:

- we know the system better, so many parameters that were once programmable (i.e., rip_byte_order) are hard-coded.
- for anything temporary (i.e., shifting the plot for testing), a command is used (<<plot shift>>) instead of changing a parameter.

All NVS parameters are loaded from saved to active RAM on power up. For head parameters, active values from the head are copied to MPE active RAM on startup

NVS parameters are local to subsystems. Subsystems can have their own NVS parameters that no other subsystems can access. All NVS parameters are treated the same, other than access restrictions. Some parameters can only be accessed when the user is in specific access modes.

Parameters can be viewed and modified by using the <<set>> command. Entering the <<set>> command alone displays a list of subsystems that have NVS parameters.

Changing Parameters

MPE-devices

When a parameter is changed using the <<set>> command, only the active RAM copy of the parameter is changed. The parameter must be saved if it is to be kept around through a power cycle. Use the <<nvs save>> command to save a single parameter or an entire subsystem. The head subsystems (*head*, *hdp*, *hdm*) have to be saved all at once – it is not possible to save individual parameters.

MCE-devices

Anytime a parameter is changed, the RAM copy is automatically changed. There is only ever one copy of NVS parameters.

NVS parameters reside in two different types of hardware: FLASH and BBRAM. Flash is a special type of memory that is slow and has a limited amount of times it can be written. NVS parameters found in FLASH are parameters that do not change often, like drum parameters or carriage parameters. BBRAM is Battery Backed-up RAM. This RAM is fast, and can be used an unlimited amount of times. Parameters in BBRAM are those that change often during device operation.

Backups

It is recommended to perform a complete backup of NVS parameters periodically, and especially before making any significant changes to the device.

MPE-devices

<<nvs dump>> command: dumps all the NVS parameters (but not state variables) to the diagnostic monitor. This dump can be captured and saved to a text file for later reference. This file is a copy only – it cannot be used to restore parameters if they get lost.

<<fsave all>> command: dumps the entire memory to the diagnostic monitor. This dump can be captured and saved to a text file for later reference. This file can be loaded back into the device with the <<fload memory>> command.

For listing out all NVS parameters, the most common commands are:

nvs dump	Displays all parameters
nvs dump diff min	Displays all parameters whose saved values are different from default. For subsystems where all parameters are default, the subsystem header is not displayed to reduce clutter

MCE-devices

To make backups of NVS parameters, both the FLASH and BBRAM memory must be saved using Service Shell. See the device's release notes on how to do this.

For listing out all NVS parameters, the most common commands are:

prm all	Displays all parameters
prm diff	Displays all parameters whose saved values are different from default. For subsystems where all parameters are default, the subsystem header is not displayed to reduce clutter.

Appendix C – Loading and Unloading Plates

For MPE-based devices, the loading and unloading was broken into sections: pick, load, unload, and eject. For MCE-based devices, there is only the concept of load and unload. Load means moving a plate from the bay or from the operator fully onto the drum. Unload means moving the plate from the drum all the way out of the machine.

Loading a plate onto the drum

MPE-based device

1. Check what's currently loaded in the bay with the command **set state bay 1**
2. If this is not correct, load the bay from the workstation, or modify ***set state bay 1***
3. Type **pick**
4. Type **load**
5. Check the status of the plate on the drum with the command **set state plate_loc d** or **d** for short

MCE-based device

1. Check what's currently loaded in the bay with the command **bay 1**
2. If this is not correct, load the bay from the workstation, or modify it with the command **bay 1 load …**
3. Type load bay 1
4. Check the status of the plate on the drum with the command **loaded**.

Unloading a plate from the drum

MPE-based device

- Type unload
- Type process

MCE-based device

- Type unload

Appendix D - Suspend Responses

The suspend responses in MCE firmware are slightly different than in MPE firmware. The main difference is the replacement of the “Clear” concept with the “Release” concept; and the addition of an “Eject option”.

Resume

Try the last step again. This operation is the same as on an MPE-based device.

Release

Tells the firmware to release the media, if possible, so it can be easily removed. This may or may not involve any actions, depending on the state of the machine. For example, if the device suspends during unloading, selecting Release may involve moving the table down and turning the vacuum cups off.

Eject

Tells the firmware to immediately unload the media as soon as it gets loaded onto the drum. This option is only valid during loading, not unloading.

Cancel (same as on an MPE-based device)

Tells the firmware to cancel the job. This operation is the same as on an MPE-based device. If a cancel is received in response to an error during loading:

- the firmware will try to back out of the load sequence if it hasn’t gone too far
- if it can’t back out, it will continue to load the plate onto the drum
- if it can’t complete the load, it will suspend again. The media may need to be removed manually from the device

If a cancel is received in response to an error during imaging, the plot stops. If pixels have been imaged, the media is unloaded. If pixels have not been imaged, the media is left on the drum for the next job.

Appendix E - Sensors

For most actuators, a solenoid is told to move, and the firmware waits until a sensor indicates the move is complete. After a set time, if the sensor still doesn't read true, then an error is reported.

In MPE, the timeout is generally an NVS parameter, the default of which has been set to a time longer then the motion should ever take under normal conditions. In MCE, the timeout is generally a hard-coded value that cannot be changed without a new firmware build.

Nearly all sensors can be disabled – this means that the sensor is not checked. The firmware will activate the actuator, then wait the full time specified by the timeout value, before continuing on. Note: when sensors are disabled, machine operation is slowed down significantly, and some critical safety checks are not done. Sensors should only be disabled as a temporary measure until the sensor can be replaced

Examples

Bypass (or not) the load ramp plate sensor

```
scan ps_loadsensor bon        bon=bypass on

scan ps_loadsensor boff       boff=bypass off
```

Show everything currently bypassed

scan bstatus

List of Sensors

Autoloader sensors (<<scan al_sensor>>)

Sensor Name	MPE name	Location
AL_SensorTableExitHome	al pes	sensor at end of unload table (home side for News; disabled for AL)
AL_SensorTableEntryAway	n/a on MPE	sensor at engine-end of unload table (away side for News; enabled for AL)
AL_SensorTableEntryHome	n/a on MPE	sensor at engine-end of unload table (home side for News; disabled for AL)
AL_SensorTableCupsDown	al ucs	unload table cups "down" sensor
AL_SensorRotatorRotate	al urs	unload table cups rotated sensor (disabled on News)
AL_SensorPlateStackHome	al pss	plate stack sensor (plates in bay) (home side for News; disabled for AL)

AL_SensorPlateStackAway	al enabled for AL)	plate stack sensor (plates in bay) (away side for News; enabled for AL)
AL_SensorCoverClosed	al pcs	plate cover closed sensor
AL_SensorPickerWristOut	al pios	picker wrist out
AL_SensorPickerOutDown	al puds	picker out and down
AL_SensorPickerOutUp	al puds	picker out and up
AL_SensorSeparatorRetract	al ss	separators out (clear of the bay)
AL_SensorTableDown	al uts	table down
AL_SensorTableUp	al uts	table up
AL_SensorTableExitAway	al pes	sensor at end of unload table (away side for News; enabled for AL)
AL_SensorRotatorRetract	al urs	unload table cups normal (unrotated) sensor (disabled on News)
AL_SensorTravellerRotate	"al prs";	optical sensor to indicate position where rotation is allowed (disabled for News)
AL_SensorTravellerHome	n/a on MPE;	cannot be bypassed
AL_SensorTravellerAway	n/a on MPE	cannot be bypassed

Engine sensors:
Trailing Edge sensors (<<scan tec_sensor>>)

Sensor Name	MPE name	Location
TEC_SensorLock	al tls	trailing edge clamp locked
TEC_SensorUnlock	al tls	trailing edge clamp unlocked
TEC_SensorUp	al tes	trailing edge clamp up (off drum) sensor
TEC_SensorAwaySideDown	al tes	trailing edge clamp down (on drum) sensor - away side
TEC_SensorHomeSideDown	al tes	trailing edge clamp down (on drum) sensor - home side
TEC_SensorClamp	al ms	trailing edge clamps detected on backbone

Roller sensors (<<scan rlr_sensor>>

Sensor Name	MPE name	Location

RLR_SensorUp	al rs	roller up (off drum)
RLR_SensorAwaySideDown	al rs	roller down (on drum) - away side
RLR_SensorHomeSideDown	al rs	roller down (on drum) - home side

Leading edge clamp sensors (<<scan lec_sensor>>)

Sensor Name	MPE name	Location
LEC_SensorUp	al les	leading edge clamp up (off drum)
LEC_SensorAwaySideDown	al les	leading edge clamp down (on drum) - away side
LEC_SensorHomeSideDown	al les	leading edge clamp down (on drum) - home side

Load ramp sensors (<<scan lr_sensor>>)

Sensor Name	MPE name	Location
LR_SensorOutTop	al lrs	load ramp out (away from drum) - top (Autoloader only)
LR_SensorOutBottom	al lrs	load ramp out (away from drum)

Exit door sensor (<<scan edr_sensor>>)

Sensor Name	MPE name	Location
EDR_SensorClosed	al es	exit door closed sensor (base/Spectrum only)

Plate sensors on load ramp (<<scan ps_load>>)

Sensor Name	MPE name	Location
PS_LoadRampAway	al lps	plate sensor on load ramp - away side (News only)
PS_LoadRamp	al lps	plate sensor on load ramp

Appendix F - More on the Scan Command

Show status of all elements beginning with letter *x*

scan p

scan t

scan l

scan d

scan a

help scan

Appendix G - Help

There are three access levels in MCE for diagnostic commands:

- field basic commands to run the system
- expert more complicated, or more dangerous commands
- dev commands put in for debugging – these should not be needed in the field.

The default access level field access. To change the access level, type

<<access field | eng | dev>>.

Typing <<help>> brings up a list of all the primary commands available on the machine, organized in alphabetical order. The help command does not have the confusing syntax of the MPE commands You can get more information on each individual command by typing <<help <command>>>. When you type <<help drum>>, you get a full list of all the drum commands available at your access level:

```
3244>> help drum

  drum                              Display drum status
  drum idle                         Idle drum (drum amp disabled)
  drum init                         Initialize drum
  drum stop                         Stop drum
  drum stopat <position>            Stop drum at position
    <position>                      <+/-degrees>
  drum spin <velocity>              Spin drum
    <velocity>                      <+/-RPM>
  drum move <position>              Position drum fast
    <position>                      <+/-degrees>
  drum move <position> [options]
    <position>                      <+/-degrees>
    [<velocity>]                    <+/-RPM>
    [s]                             shortest distance
  drum smove <position>             Position drum, allow settling
    <position>                      <+/-degrees>
  drum smove <position> [options]
    <position>                      <+/-degrees>
    [<velocity>]                    <+/-RPM>
    [s]                             shortest distance
```

Appendix H - Verbosity

Knowledge of tasks is not necessary for operating the machine, but it is useful for understanding why some things are done the way they are. The verbose levels are one example. Each task has its own verbose setting. Most lines in the diagnostic output are preceded by the task name.

Most lines in the diagnostic monitor include the verbosity. The verbosity level is one of the following C, S, E, or D, corresponding to Customer, Service, Engineer and Developer. The verbosity level means that the text is only displayed at the specified verbosity level or higher. The task name is the three-letter acronym given in the second column of the example below. See also the table at the end of this section for an explanation of the

Examples:

D	MDH:	Command sent properly
D	PRD:	PRD NVS Callback, NVS parameter has changed: d3
E	DRS:	Drum command received 4704

To change the verbosity level of the display use one of the following commands:

- <<vbs user customer>>, least amount of information
- <<vbs user service>>
- <<vbs user eng>>
- <<vbs user debug>>, most amount of information

Devices usually default the user level to customer, therefore providing the least amount of information. For debugging, set the verbosity to service <<vbs user service>>. The user level is always reset on a device reset.

NOTE: by using user levels such as engineering or debug the firmware may provide a lot of information and it WILL slow down the device. So only uses these for debugging problems and not as a common setting.

For a list of all subsystems, the command <<vbs>> will display most of the subsystems using their three-letter acronym.

Table 1: Subsystems common to all MCE-based devices

Task Name	Primary function
ERR	Error subsystem, this is used for firmware error handling.
NUC	This is the main MCE operating system task, which is called Nucleus.
MOP	This is the monitor interface-parsing task. It handles diagnostics monitor interface input.
UTL	A general utility task.
VBS	The verbosity task, it controls what is displayed on the diagnostics monitor.
MON	The diagnostics monitor task.

Task Name	Primary function
DRS	Drum control task.
CRG,CAR	Carriage control task
MPC	Plot controller task.
MIC	Image controller task
376	Microprocessor controller task.
MSG	Message controller task.
PRM	NVS parameters task.
MDH	Media handler task.
PRD	Product specific task.